AI '92

MAY 11-15

UNIVERSITY OF BRITISH COLUMBIA
VANCOUVER, BRITISH COLUMBIA

Proceedings Ninth Canadian Conference on Artificial Intelligence

Actes neuvième conférence canadienne sur l'intelligence artificielle

Edited by/Éditée par:
Janice Glasgow and
Robert Hadley

Sponsored by:
Canadian Society for Computational Studies of Intelligence

Commanditée par:
Société Canadienne pour l'Étude de l'Intelligence par Ordinateur

Proceedings of the Ninth Biennial Conference of the Canadian Society for Computational Studies of Intelligence

Actes de la Neuvième Conférence Biennale de la Société Canadienne pour l'Étude de l'Intelligence par Ordinateur

edited by/sous la direction de Janice Glasgow and/et Robert F. Hadley

University of British Columbia,
Vancouver, British Columbia, Canada
11-15 May, 1992

Sponsored by/ Parrainée par
Canadian Society for Computational Studies of Intelligence

Supported by/ Avec l'appui financier de
The BC Advanced Systems Institute
Bell-Northern Research Limited
Centre for Systems Science, Simon Fraser University
Information Technology Research Centre of Ontario
The Manufacturing Research Corporation of Ontario
Natural Sciences and Engineering Research Council of Canada
PRECARN Associates Inc.
Industry, Science, and Technology of Canada

In Cooperation with/ Et la collaboration de
University of British Columbia
Canadian Human-Computer Communication Society
Canadian Image Processing and Pattern Recognition Society

ISBN 0-9694596-1-0

Canadian Society for Computational Studies of Intelligence
Société Canadienne pour l'Étude de l'Intelligence par Ordinateur

Edited by/Sous la direction de
Janice Glasgow and/et Robert F. Hadley

Imprimé par Pro Printers B.C. Ltd.

Within Canada, copies of these proceedings may be obtained as follows. Send orders, together with payment of: $35 CDN each (CSCSI members), $40 CDN each (non-members) (Add $5 CDN for postage) to:

CIPS
243 College Street (5th floor)
Toronto, Ontario M5T 2Y1
CANADA

Outside of Canada, please contact:

Morgan Kaufman Publishers Inc.
Order Fulfillment Center
P.O. Box 50490
Palo Alto, California 94303
USA

Au Canada, on peut obtenir des exemplaires des présents actes en procédant comme suit. Adressez les demandes accompagnées d'un paiement de: 35 $ can. chacun (membres de la SCEIO), 40 $ can. chacun (non-membres) (Ajouter 5 $ can. pour frais de poste) à:

ACI
243, rue College (5e étage)
Toronto, Ontario M5T 2Y1
CANADA

En dehors du Canada, adressez les demandes à:

Morgan Kaufman Publishers Inc.
Order Fulfillment Center
P.O. Box 50490
Palo Alto, California 94303
USA

CSCSI '92 Program Committee
Comité du Programme SCEIO '92

Program Chairs / Présidents du Comité de Programme

Janice Glasgow
Department of Computing and Information Science
Queen's University

Bob Hadley
School of Computing Science
Simon Fraser University

Program Committee / Comité de Programme

Veronica Dahl
Simon Fraser University
Renato De Mori
McGill University
Vasant Dhar
New York University
Brian Funt
Simon Fraser University
Randy Goebel
University of Alberta
Russell Greiner
Siemens Corporate Research
Rob Holte
University of Ottawa
Larry Hunter
National Library of Medicine
Alan Mackworth
University of British Columbia
Mary McLeish
University of Guelph
Bob Mercer
University of Western Ontario
John Mylopoulos
University of Toronto
Eric Neufeld
University of Saskatchewan
Peter Patel-Schneider
AT&T
Dick Peacocke
Bell Northern Research
David Scuse
University of Manitoba

Referees / Arbitres

Yawar Ali
John Anderson
Jamie Andrews
Fahiem Bacchus
Allan Bennett-Brown
Philippe Besnard
Alex Borgida
Michel Boyer
Gordon Brown
Peter Caines
Vinay Chandra
Jerome Chiabaut
Lawrence Chung
Peter Clark
Darrell Conklin
Julian Craddock
Prem Devanbu
Michel Desmarais
A.K. Dewdney
Judy Dick
George Drastal
Mark Drew
Denys Duchier
Renee Elio
Mark Evans
Dan Fass
Michel Feret
Innes Ferguson
Suryanil Ghosh
K. Glover
Scott Goodwin
Chris Groeneboer
Michael Gruninger
Ranabir Gupta
K. Hadavi
Gary Hall
Jia Wei Han
Bill Havens
Yong Hu
Hing-Kai Hung
Hardeep Johar
S. Judd
Kenji Kanazawa
Henry Kautz
Mike Kelly
Jan Komorowski
Brian Kramer
Roland Kuhn
Guy Lapalme
Yves Lesperance
Tim Lethbridge
Ze-Nian Li
Dekang Lin
Charles Ling
David Lowe
Raghav Madhavan
A.A. J. Marley
Stan Matwin
Parham Momtahan
Jian-Yun Nie
Brian Nixon
Yves Normandin
Bill Older
Franz Oppacher
Russell Ovans
Dimitris Plexousaki
Fred Popowich
Bob Price
Carlos Saldanha
Jacques Savoy
Kaushik Saroj
Dale Schurmans
Marek Sergot
Gregory Sidebottom
Fei Song
Frank Tong
Thodoros Topaloglou
Peter Van Beek
Jean Vaucher
Andre Vellino
Chenghui Wang
Huaiqing Wang
Phil Winne
Michael Wong
Jia-Huai You
Li Yan Yuan
Ying Zhang

Message from the Chairs

This volume comprises the Proceedings of the Ninth Biennial Conference of the Canadian Society for Computational Studies of Intelligence. Over the last 17 years, biennial conferences sponsored by CSCSI have acquired a reputation for excellence, collegiality and friendliness. It is our belief that this conference continues the tradition, and that the volume now before you testifies to the high quality and timeliness of research presented to the conference.

This year 112 papers were submitted in time to be considered for the conference. Of these, 35 were selected for presentation and inclusion in these proceedings. Each of these papers was reviewed by at least two reviewers, and in some instances, as many as four were consulted. Undoubtedly, some worthy papers were not accepted, due to the number of papers submitted and the time constraints of our program. Final decisions were made after careful deliberation by six members of program committee, including the program co-chairs.

We would like to take this opportunity to recognize and thank the people who made this program and conference possible. First, we would like to thank our invited, distinguished speakers, who have contributed much to the quality and success of the meeting. We would also like to thank all members of our program committee and auxiliary review committee, who contributed long hours and much effort to make this program a success. Recognition is due to the general chairs of the three conferences, Kellogg Booth and Alain Fournier, our publicity chair, Fred Popowich and our local arrangements chair, David Poole. We would also like to express our appreciation to Carol Morrison, Sandra Crocker, Christine Adams, Ranabir Gupta, Pierre Massicotte, Michel Feret, members of the Centre for Systems Science (at Simon Fraser University) and of MAGIC (at the University of British Columbia) for their administrative, technical and French translation assistance in the organization of the conference and the production of these proceedings. In addition, we are indebted to Peter Patel-Schneider, Dick Peacocke, and Nick Cercone for their continual and invaluable advice and guidance in the organization of this conference.

We wish you all an enjoyable and rewarding conference.

Janice Glasgow and Bob Hadley
Program Co-Chairs, AI '92

Message des Présidents

Ce volume contient les Actes de la Neuvième Conférence Biennale de la Société Canadienne pour l'Etude de l'Intelligence par Ordinateur. Durant les 17 dernières années, les conférences biennales ont acquis une réputation d'excellence, de collégialité et d'ouverture. Nous pensons que la présente édition de cette conférence maintient cette tradition. Ce volume, maintenant entre vos mains, témoigne de la haute qualité et de la pertinence des recherches présentées à cette conférence.

Cette année, 112 communications furent envoyées dans les délais impartis. Trente cinq furent retenues pour présentation à cette conférence et sont publiées dans ces actes. Chacun de ces articles a été jugé par au moins deux arbitres, et dans certains cas, par pas moins de quatre. Sans aucun doute, certaines communications de valeur n'ont pas été acceptées. Ceci est dû au grand nombre d'articles soumis et aux contraintes de temps liées à notre programme. Les décisions finales ont été prises après délibération attentive d'un jury de six membres du comité de programme, incluant les présidents de la conférence.

Nous tenons à exprimer nos remerciements aux personnes qui ont participé à la mise sur pied de cette conférence: aux conférenciers invités, dont la contribution à la qualité et au succès de la conférence a été grandement appréciée, aux membres du comité de programme et du sous-comité d'arbitrage, qui ont travaillé de longues heures à l'élaboration du programme, aux présidents des trois conférences, Kellogg Booth et Alain Fournier, au directeur de la publicité, Fred Popowitch, ainsi qu'au directeur pour l'organisation locale, David Poole. Nous tenons aussi à exprimer notre gratitude à Carol Morrison, Sandra Crocker, Christine Adams, Ranabir Gupta, Pierre Massicotte, Michel Féret, aux membres du Centre pour la Science des Systèmes (Université Simon Fraser), et à MAGIC (Université de Colombie Britannique) pour l'assistance administrative, technique et de traduction qu'ils ont fournie pour l'organisation de cette conférence et pour la publication de ces actes. Finalement, nous sommes redevables à Peter Patel-Schneider, Dick Peacocke, et à Nick Cercone de leurs précieux conseils et de leur aide assidue pour l'organisation de cette conférence.

Nous vous souhaitons, à tous, une conférence productive et agréable.

Janice Glasgow et Bob Hadley
Présidents du comité de programme, IA 92

CSCSI '92 Organizing Committee
Comité Organisateur SCEIO '92

General Chairs / Présidents Generaux

Kellogg Booth
University of British Columbia

Alain Fournier
University of British Columbia

Program Chairs / Présidents du Comité de Programme

Janice Glasgow
Queen's University

Robert F. Hadley
Simon Fraser University

Local Arrangements Chair / Organisation Locale

David Poole
University of British Columbia

Publicity Chair / Publicité

Fred Popovich
Simon Fraser University

Invited Speakers/ Conférenciers Invités

Alan Mackworth, University of British Columbia
Using Constraints

Alan Bundy, University of Edinburgh
How To Prove Theorems by Induction

Don Perlis, University of Maryland
Memory, Mind, and Models of Self

Veronica Dahl, Simon Fraser University
What Linguistics Can Contribute to AI

David Waltz, Thinking Machines Inc. and Brandeis University
AI and Massive Parallelism

CSCSI Executive Committee 1990-1992
Exécutifs de la SCEIO 1990-1992

President/ Président

Ian Witten
Head of Computer Science
University of Calgary

Past President/ Président Précédent

Dick Peacocke
Bell-Northern Research

Vice-President/ Vice-Président

Janice Glasgow
Department of Computing and Information Science
Queen's University

Secretary/ Secrétaire

Peter Patel-Schneider
AT&T Bell Laboratories

Treasurer/ Trésorier

Russ Thomas
National Research Council

Editor/ Editeur

Roy Masrani
Alberta Research Council

Best Paper Award

The CSCSI best paper award is sponsored by the Editorial Board of *Artificial Intelligence*. It is given for the paper or papers that best combine significant new results with clarity of writing and accessibility across the conference. The sponsorship by the board provides both an honorarium and a rapid review process in the journal for an extended version of the conference paper(s).
This year the CSCSI Program Committee is pleased to award the CSCSI best paper award to Russell Greiner, for "Probabilistic Hill-Climbing: Theory and Applications".

Prix de la Meilleure Communication

Le prix CSEIO de la meilleure communication est parrainé par le conseil de rédaction de la revue *Artificial Intelligence*. Le prix est décerné à la ou à les communications qui allient au mieux l'importance des résultats, la clarté de l'expression, et l'accessibilité au plus grand nombre. Le parrainage du conseil comprend un prix en espeèces et une procédure d'évaluation rapide, en vue de la publication dans la revue de versions étendues des communications.
Le comité de programme de la conférence SCEIO 1990 est heureux de décerner le prix de la meilleure communication à Russell Greiner, pour "Probabilistic Hill-Climbing: Theory and Applications".

CSCSI Distinguished Service Award

1992 - John Mylopoulos

The executive of the Canadian Society for Computational Studies of Intelligence (CSCSI/SCEIO) is pleased to announce that John Mylopoulos of the University of Toronto will be presented with the inaugural CSCSI Distinguished Service Award. Henceforth, this prestigious award will be presented biennially to an individual who has made outstanding contributions to the Canadian AI community in one or more of the following areas: community service, research, training of students, and research/industry interaction. John Mylopoulos is considered by many to be the "father" of AI research in Canada. One of his most important accomplishments is the supervision of many of Canada's AI PhD and MSc students, several of whom have gone on to have significant careers. John is also responsible for the birth, nurturing and adolescence of the AI group we see today at the University of Toronto.

John's public service accomplishments are a matter of record and include serving on the steering committee for the formation of CSCSI to more recently co-chairing the IJCAI'91 conference. He either is or has been an important member of virtually every AI-oriented academic organization in Canada, including: Senior Fellow, CIAR; CIAR/PRECARN Associate; and Principal Investigator in ITRC, IRIS, and PRECARN's APACS project.

John Mylopoulos has also made significant contributions in the area of research in AI. His work in knowledge representation systems involves formalisms for integrating concepts from semantic networks, logical and procedural representations and has resulted in two systems, PSN (Procedural Semantic Networks) and Telos. His research in the application of knowledge representation systems to information system development has produced a series of requirements modelling and design languages culminating in Taxis, intended for the design of interactive information systems.

Prix du Mérite de la SCEIO

1992 - John Mylopoulos

Le Conseil exécutif de la Société Canadienne pour l'Etude de l'Intelligence par Ordinateur (SCEIO/CSCSI) a l'honneur de décerner le Prix inaugural du Mérite de la SCEIO à John Mylopoulos de l'université de Toronto. Cette récompence prestigieuse sera dorénavant remise biannuellement à une personnalité qui aura fait bénéficier la communauté canadienne de l'IA de contributions remarquables dans un ou plusieurs des domaines suivants: service rendu à la communauté, recherche, enseignement, rapports recherche-industrie. John Mylopoulos est considéré par beaucoup comme le "père" de la recherche en IA au Canada. Ses qualités de superviseur ont permis à de nombreux étudiants au doctorat et en maîtrise d'entreprendre de carrières brillantes. John Mylopoulos a littéralement forgé le groupe d'IA de l'université de Toronto, pour l'amener au niveau d'excellence que la communauté lui reconnaît aujourd'hui.

Les services rendus par John Mylopoulos à la communauté sont bien connus, notamment sa participation au comité constitutif de la SCEIO et, plus récemment, son poste de vice-président de la Conférence IJCAI'91. Il est, ou a été, une personnalité d'importance au sein de nombreux organismes académiques orientés vers l'IA: Membre de la CIAR, Membre Associé de la CIAR/PRECARN, et Directeur de recherche de projets ITRC, IRIS et PRECARN-APACS.

Les contributions de John Mylopoulos dans le domaine de l'IA sont importantes. Son travail relatif aux systèmes de représentation de connaissances a abouti à des formalismes d'intégration de concepts provenant des réseaux sémantiques, ainsi que des représentations logiques et procédurales. Ces travaux ont permis le développement de deux systèmes que sont PSN (Procedural Semantic Network) et Telos. Ses recherches traitant de l'application des systèmes de représentation de connaissances au développement de systèmes d'information ont produit une série de langages de modélisations de spécifications et de design tels Taxis, orientés vers la conception de systèmes interactifs d'information.

Table of Contents / Table des Matières

Planning / Plannification

Philosophical Issues and Data Classification / Issues Philosophiques et Classification de Données

Search / Fouille

Applications / Applications

Reasoning with Uncertainty / Raisonnement avec Incertitude

Default Reasoning / Raisonnement par Défaut

Constraint-Based Reasoning / Raisonnement à Base de Contraintes

Knowledge Representation / Représentation de Connaisances

Learning / Apprentissage

PANEL DISCUSSION / TABLE RONDE

PANEL: An Interdisciplinary View of Constraint Reasoning (Chair: Veronica Dahl)
Participants:

Veronica Dahl, Simon Fraser University
Bill Havens, Simon Fraser University
Alan Mackworth, University of British Columbia
Greg Sidebottom, Simon Fraser University
Pascal van Hentenryck, Brown University

Generating Object Descriptions: Integrating Examples with Text

Vibhu O. Mittal[†*] and **Cécile L. Paris**[*]

[*]USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
U.S.A

[†]Department of Computer Science
University of Southern California
Los Angeles, CA 90089
U.S.A

Abstract

Descriptions of complex concepts often use examples for illustrating various points. This paper discusses the issues that arise in generating complex descriptions in tutorial contexts. Although some tutorial systems have used examples in explanations, they have rarely been considered as an integral part of the complete explanation – they have usually been merely supportive devices – and inserted in the explanations without any representation in the system of how the examples relate to and complement the textual explanations that accompany the examples. This can lead to presentations that are at best, weakly coherent, and at worst, confusing and mis-leading for the learner.

In this paper, we consider the generation of examples as an integral part of the overall process of generation, resulting in examples and text that are smoothly integrated and complement each other. We address the requirements of a system capable of this, and present a framework in which it is possible to generate examples as an integral part of a description. We then show how techniques developed in Natural Language Generation can be used to build such a framework.

1 Introduction

It has long been known that examples are very useful in communication – especially in explanations and instruction. New ideas, concepts or terms are conveyed with greater ease and clarity, if the descriptions are accompanied by appropriate examples (e.g., [Houtz *et al.*, 1973; MacLachlan, 1986; Pirolli, 1991; Reder *et al.*, 1986; Tennyson and Park, 1980]). People like examples because examples tend to put abstract, theoretical information into concrete terms they can understand.

We gratefully acknowledge the support of NASA-Ames grant NCC 2-520 and DARPA contract DABT63-91-C-0025. The authors may be contacted through electronic mail at: {MITTAL,PARIS}@ISI.EDU

Explanation capabilities are becoming increasingly important nowadays, especially as domain models become larger and more specialized. One requirement in such systems is the ability to explain complex objects, relations or processes that are represented in the system. Advances in natural language generation and research in user modelling have resulted in impressive descriptions being produced by such systems. However, these systems have not, for the most part, concentrated on the issue of generating examples as a part of the overall description. While examples can, in some cases, be *retrieved* from a pre-defined 'example-base' and added to the description, using examples effectively, as an important and a complementary part of the overall description, requires the system to reason with the constraints introduced by both the textual explanation, as well as the examples, in making decisions during the generation process.

There are many issues that must be considered in selecting and presenting examples – in this paper, we shall briefly highlight some of these issues. We view example generation as an integral part of generating descriptions, because examples affect not only other examples that follow, but also the surrounding text. We describe how a text generation system that *plans* text in terms of both intentional and rhetorical goals, can be structured to plan utterances that can include examples in an integrated and coherent fashion. Some of the issues addressed in this regard are equally important in the planning and presentation of *other* explanatory devices – such as diagrams, pictures and analogies.

2 Previous Work and Unaddressed Issues

Most previous approaches to the use of examples in generating descriptions and explanations focused on the issue of *finding* useful examples. Rissland's (1981) CEG system , for instance, investigated issues of retrieval versus construction of examples; Rissland and Ashley's (1986) HYPO system retrieved examples and investigated techniques for modifying them along multiple dimensions to fit required specifications; Suther's example generator [Suthers and Rissland, 1988] is also similar to CEG, and investigated efficiency of search techniques in finding examples to modify. Later work by Woolf and her colleagues focused on design issues of tutoring systems, including determination of when examples are necessary [Woolf and McDonald, 1984;

Woolf and Murray, 1987]. However, it did not address issues of discourse generation and the integration of text with examples.

Our work builds upon these and other studies (e.g., [Reiser *et al.*, 1985; Rissland *et al.*, 1984; Woolf *et al.*, 1988]) to study how to provide *appropriate, well-structured and coherent examples in the context of the surrounding text*. It has been shown previously that presentation of descriptions without the use of examples is not effective, perhaps because the use of definitions on their own may lead to a learner merely memorizing a string of verbal associations [Klausmeier, 1976]. The converse – presentation of examples alone – has also been shown previously to be less effective than the use of both examples and descriptions; the use of both almost doubled user comprehension in certain cases (e.g., [Charney *et al.*, 1988; Feldman, 1972; Feldman and Klausmeier, 1974; Gillingham, 1988; Klausmeier, 1976; Merrill and Tennyson, 1978; Reder *et al.*, 1986]). However, text and examples that are not well integrated can cause greater confusion than either one alone (e.g., [Ward and Sweller, 1990]). Thus it is clear that if descriptions are to make use of examples effectively, both the descriptions and the examples must be integrated with each other in a coherent and complementary fashion. Furthermore, there is often a lot of *implicit information* in the sequence of presentation of the examples. The system should be capable of representing this information to structure the sequence correctly.

There are many points that must be considered by any system that attempts to generate effective descriptions of complex concepts:

1. What aspects of information should be exemplified? A system is likely to have detailed knowledge about the concept. It typically needs to select some aspects to present to the user. (This issue has often been raised in natural language generation.)
2. When should a description include examples? A few researchers have started to look at this issue [Woolf and McDonald, 1984; Woolf and Murray, 1987; Woolf *et al.*, 1988], although much work still remains to be done.
3. How is a suitable example found? Is it retrieved from a pre-defined knowledge base and modified to meet current specifications (as in [Rissland *et al.*, 1984]), or is it constructed (as in [Rissland, 1981])?
4. How should the example be positioned with respect to the surrounding text? Should the example be *within* the text, *before* it, or *after* it?
5. Should the system use one example, or multiple examples? If more than one example is used, how many should be used, and what information should each one convey?
6. If multiple examples are to be presented, how is the order of presentation to be determined?
7. What information should be included in the prompts[1] and how can they be generated?

[1] 'Prompts' are attention focusing devices such as arrows, marks, or even additional text associated with examples [Engelmann and Carnine, 1982].

A list always begins with a left parenthesis. Then come zero or more pieces of data (called the elements of a list) and a right parenthesis. Some examples of lists are:

```
(AARDVARK)                  ;;; an atom
(RED YELLOW GREEN BLUE)     ;;; many atoms
(2 3 5 11 19)               ;;; numbers
(3 FRENCH FRIES)            ;;; atoms & numbers
```

A list may contain other lists as elements. Given the three lists:

(BLUE SKY) (GREEN GRASS) (BROWN EARTH)

we can make a list by combining them all with a parentheses.

((BLUE SKY) (GREEN GRASS) (BROWN EARTH))

Figure 1: A description of the object LIST using examples (From [Touretzky, 1984], p.35)

---

8. How is lexical cohesion maintained between the example and the text? The text and the example(s) should probably use the same lexical items to refer to the same concepts.
9. We already know from work in natural language generation that a description is affected by issues such as user-type, text-type, etc. How is the generation of examples (when they should be included, the type of information that they illustrate, and the number of examples) within a text affected by:
   - the prospective audience-type (naive vs. advanced, for instance),
   - the knowledge-type (concepts vs. relations vs. processes),
   - the text-type (tutorial vs. reference vs. report, etc), and
   - the dialogue context?

While each of these issues needs to be addressed in a practical system, we shall only discuss some of them here: the issues of positioning the example within a text, the number of examples to be presented, and their order. (Issues #1, #2 and #3 have already been studied to some extent, by other researchers, for e.g., [Paris, 1987; Woolf *et al.*, 1988; Rissland *et al.*, 1984; Rissland and Ashley, 1986; Rissland, 1983]). We now discuss in more detail, the points we are concerned with. We illustrate each point with the description in Figure 1. We are not yet (for this paper) addressing the issues of user-type, the text-type and the dialogue context, though they can all affect the generation of examples. We take as our initial context the generation of a description in a tutorial fashion, for a naive user and as a 'one-shot' response.

3 Integrating Examples in Descriptions

As mentioned previously, a number of studies have shown the need for examples to illustrate descriptions and definitions, as well as a need for explanations to complement the examples

presented. Presentation of either on their own is not as useful an approach as one that combines both of them together.

3.1 Positioning the Example and the Description:

Should the example be placed *before*, *within* or *after* the accompanying text? This is an important issue, as it has been reported that there are significant differences that can result from the placement of examples before and after the explanation [Feldman, 1972].

Our analyses of different instructional materials reveal that examples usually tend to follow the description of a concept in terms of its critical attributes. Critical attributes are attributes that are *definitional* – the absence of any of these attributes causes an instance to not be an example of a concept. In the lisp domain, for example, a LIST has parentheses as its critical attributes; the elements of the list itself are not. The examples can then be followed by text which elaborates on features in the examples, unless prompts are included with the examples. If more than one example needs to be presented, the example is usually placed separately from the text (rather than within it). Otherwise, the example is integrated within the text, as in: `An example of a string is "The quick brown fox"`. Following the examples, the description continues with other attributes of the concept, possibly accompanied by further examples.

Sometimes, examples are used as elaborations for certain points which might otherwise have been elaborated upon in the text. For instance, in Figure 1, the LIST could have been described as "A list always begins with a left parenthesis. Then come zero or more pieces of data, *which can be either symbols, numbers, or combinations of symbols and numbers*, followed by a right parenthesis." Instead, the elaboration on the data types is embodied in the information present in the examples. The first set of examples in Figure 1 have prompts associated with them, highlighting features (number and type information) about the examples. Following these examples, some text elaborates on the fact that the elements of a list can also be lists. Further examples of lists are used to show how these can be combined to form another list.

3.2 Providing the Appropriate Number of Examples

Studies have indicated that information transfer is maximized when the learner has to concentrate on as few features as possible [Ward and Sweller, 1990]. This implies that teaching a concept is most effectively done one feature at a time. This has important implications for example generation: it indicates that examples should try and convey one point at a time, especially if the examples are meant to teach a new concept. Thus, should the concept have a number of different features, a number of examples are likely to be required, one (or a set of) examples for each feature. This is also supported by experiments on differences in learning arising from using different numbers of examples [Clark, 1971; Feldman, 1972; Klausmeier and Feldman, 1975; Markle and Tiemann, 1969].

This is illustrated in Figure 1, in which each example highlights one feature of LISTs: that the data can be a single symbol, a number of symbols, numbers, etc. Contrast those examples, with a single example which summarizes most of the features a LIST can have, as given below:

```
(FORMAT T "~ A~ A~ A" 'abcdef 123456
          '(abc (123 ("ab"))) ( ))
```

It is important that the system generate an appropriate number of examples, each emphasizing certain selected features.

3.3 Ordering the Examples

Given a number of examples to present, the sequencing is also an important matter, because examples often build upon each other. Furthermore, the difference between two adjacent examples is significant, as proper sequencing can be a very powerful means of focusing the hearer's attention (e.g., [Feldman, 1972; Houtz *et al.*, 1973; Klausmeier *et al.*, 1974; Litchfield *et al.*, 1990; Markle and Tiemann, 1969; Tennyson *et al.*, 1975; Tennyson and Tennyson, 1975]). Consider the sequence of examples on LISTs in Figure 1: the first two examples focus attention on the *number* of elements in a LIST – they highlight the fact that a LIST can have any number of elements in it; the second and third ones illustrate that symbols are not always required in a LIST – a LIST can also be made up of numbers; the fourth example contrasts with the third, and illustrates the point that a LIST need not have elements of just one type – both numbers as well as symbols can be in a LIST at the same time.

It has also been shown that presenting easily understood examples before presenting difficult[2] examples has a significant beneficial effect on learner comprehension [Carnine, 1980]. Ordering is thus important – it is worth noting that the linguistic notion of the *maxim of end-weight* [Giora, 1988; Werth, 1984], also dictates that difficult and new items should be mentioned after easier and known pieces of information; since there is a direct correlation between the description and the examples, this maxim offers additional motivation for a sequencing of the examples from easy to difficult. Possible orderings may also depend upon factors such as the type of concept being communicated (whether for instance, it is a disjunctive or a conjunctive concept) or whether it is a relation. For example, in Figure 1, the order of examples is determined both by the order in which features are mentioned ("zero or more" and "pieces of data"), and the complexity of examples within each grouping (symbols, followed by numbers, followed by combinations of symbols and numbers).

3.4 Generating Prompts for the Examples

Instructional materials that include examples often have tag information associated with each example. This is often referred to as "prompting" information in educational literature (e.g., [Engelmann and Carnine, 1982]). Prompts help focus attention on the feature being illustrated. They often replace long, detailed explanations, and therefore play a role similar to the one of explanation of the examples. However, as they

[2]The terms 'easy' and 'difficult' are difficult to specify, and are usually highly domain specific – in the case of LISP, for instance, one measure of difficulty is the number of different grammatical productions that would be required to parse the construct.

occur in the same sequence as the examples, they capture the change in the examples very efficiently. Consider the example sequence in Figure 1. In this case, the prompts (tags such as "List of Symbols" and "List of combined Symbols and Numbers") cause the learner to focus on the feature that is being highlighted by the examples.

3.5 Maintaining Lexical Cohesion between the Text and the Examples

In any given domain, there are likely to be a number of terms available for a particular concept. It is important that the system use consistent terminology throughout the description: both in the definition, as well as in the examples. Consider for instance, the description in Figure 1. Both the examples and the definition use the term 'list', although the terms *list*, *s-expression* and (sometimes) *form* are interchangeable. Difference in terminology can result in confusing messages being communicated to the learner (e.g., [Feldman and Klausmeier, 1974]). This issue becomes especially important in cases where the examples are retrieved, as the terms used in the example may be different from the terms used in other examples, or the definition. The construction of the textual definition and the examples must thus be done in a coordinated and cooperative manner.

3.6 The Knowledge-Type and its Effect on Descriptions

It has been observed that the type of the knowledge communicated (concept vs. relation vs. a process) has important implications for the manner in which this communication takes place (e.g., [Bruner, 1966; Engelmann and Carnine, 1982]). Not only is the information different, but the type of examples and their order of presentation is affected. This is because, if, for instance, the system needs to present information on a relation, it must first make sure that the concepts between which the relation holds are understood by the hearer – this may result in other examples of the concepts being presented before examples of the relation can be presented. The order is usually different too and there are no negative[3] examples of relations presented in initial teaching sequences (e.g., [Engelmann and Carnine, 1982; Bruner, 1966]).

In this section, we have described in somewhat greater detail, a few of the issues that we identified in Section 2. In the following section, we describe a framework for generation that addresses some of the above issues. We should mention that our system has only a simple user-model, and in the description, we shall not discuss other aspects of the system such as how dialogue is handled [Moore, 1989a], the effect of the text-type, etc. The description is meant to convey a flavor of how the system processes a goal to describe concepts and uses examples to help achieve its goal.

[3] 'Positive' examples are instances of the concept they illustrate; 'negative' examples are those which are *not* instances of the concept being described.

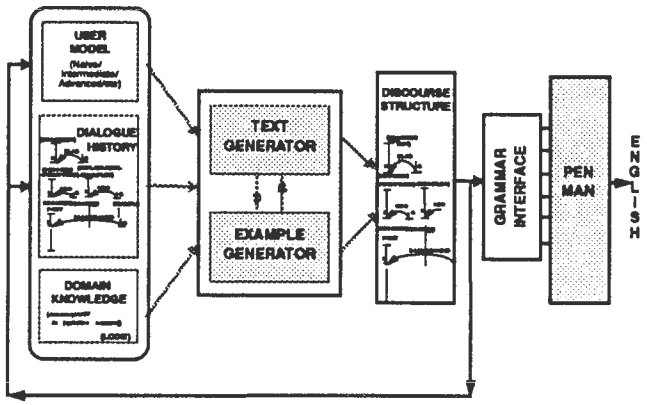


Figure 2: A block diagram of the overall system.

4 A Framework to Generate Descriptions with Examples

Using techniques developed in natural language generation, we are working on a framework within which it is possible to integrate examples in a description. This framework will also enable us to investigate and test more carefully the issues raised in Section 2, incorporating and building upon the work of other researchers (e.g., [Paris, 1991b; Woolf *et al.*, 1988; Rissland *et al.*, 1984; Rissland *et al.*, 1984; Rissland *et al.*, 1984]). Our current framework implements the generation of examples within a text-generation system by explicitly posting the goals of providing examples. Our system uses a planning mechanism: given a top level communicative goal (such as `(DESCRIBE LIST))`, the system finds plans capable of achieving this goal. Plans typically post further sub-goals to be satisfied, and planning continues until primitive speech acts – i.e., directly realizable in English – are achieved. The result of the planning process is a discourse tree, where the nodes represent goals at various levels of abstraction (with the root being the initial goal, and the leaves representing primitive realization statements, such as `(INFORM ...)` statements. In the discourse tree, the discourse goals are related through coherence relations. This tree is then passed to a grammar interface which converts it into a set of inputs suitable for input to a natural language generation system (Penman [Mann, 1983]). A block diagram of the system is shown in Figure 2.

Plan operators can be seen as small schemas (scripts) which describe how to achieve a goal; they are designed by studying natural language texts and transcripts. They include conditions for their applicability. These conditions can refer resources like the system knowledge base (KB), the user model, or the context (including the dialogue context). A complete description of the generation system is beyond the scope of this paper – see [Moore and Paris, 1992; Moore, 1989b; Moore and Paris, 1991; Paris, 1991a; Moore and Paris, 1989] for more details.

We are adding an example generator to this generation system. Examples are generated by explicitly posting a goal within the text planning system: i.e., some of the plan oper-

ators used in the system include the generation of examples as one of their steps, when applicable. This ensures that the examples embody specific information that either illustrates or complements the information in the accompanying textual description. It is clear that there are additional constraints (for e.g., the user model, text type, dialogue context, etc) that will be needed in any comprehensive implementation of example generation, but we shall investigate those issues in future work. These additional sources of knowledge can be currently added to the system by incorporating additional constraints in the plan operators which reference these resources. Thus, experimenting with different sources in an effort to study their effects is not very difficult.

The number of examples that the system needs to present is determined by an analysis of the features that need to be illustrated. These features depend on the representation of the concept in the knowledge base and the user model. Not all the features illustrated in the examples may be actually mentioned in the text. This is because the description may actually leave the elaboration up to the examples rather than doing it in the text. In Figure 1, for instance, the different data types (numbers, symbols or combinations of both) that may form the elements of a list are not mentioned in the text, but are illustrated through examples. The user model influences the choice of features to be presented. The number of examples is directly proportional to the number of features – in case of the naive user, there is usually one example per feature. In our framework, the features to be presented are determined based on the domain model and a primitive categorization of the user (naive vs. advanced).

The order of presentation of examples is dependent mainly upon the order of the features being mentioned in the text. In case the text does not explicitly mention the features (as in Figure 1, where the different data types are not mentioned in the text), the system orders them in increasing complexity. Since the ordering in the text is in an increasing order of complexity too (the maxim of end-weight), the least complex examples are presented first. We have devised domain specific measures of complexity. In the case of LISP for instance, the complexity of a structure is measured in terms of the number of different productions that would need to be invoked to parse the example.

The system maintains lexical cohesion by replacing all occurrences of equivalent terms with one uniform term. This is done as the last step in the discourse tree, before it is used as input to the language generator. There are clearly more issues to be studied to obtain lexical cohesion, but this indicates our framework's ability to at least ensure a consistent use of vocabulary. Our framework is thus centered around a text-planner that generates text and posts explicit goals to generate examples that will be included in the description. Plans also indicate how and when to generate the prompt information. By appropriately modifying the constraints on each plan-operator, we can investigate the effects of different resources in the framework. In the following section, we shall illustrate the working of the system by generating a description similar to the one in Figure 1.

4.1 A Trace of the System

The system initially begins with the top-level goal being given as `(DESCRIBE LIST)`. The text planner searches for applicable plan operators in its plan-library, and it picks one based on the applicable constraints such as the user model (introductory), the knowledge type (concept), the text type (scientific), etc. The user model restricts the choice of the features in this case (naive user) to syntactic ones. The main features of LIST are retrieved, and two subgoals are posted: one to list the critical features (the left parenthesis, the data elements and the right parenthesis), and another to elaborate upon them.

At this point, the discourse tree has only two nodes: the initial node of `(DESCRIBE LIST)` – namely `LIST-MAIN-FEATURES` and `DESCRIBE-FEATURES`, linked by a rhetorical relation, `ELABORATE`.[4]

The text-planner now has these two goals to expand:

```
LIST-MAIN-FEATURES
DESCRIBE-FEATURES
```

The planner searches for appropriate operators to satisfy these goals. The plan operator to describe a list of features indicates that the features should be mentioned in a sequence. Three goals are appropriately posted at this point. These goals result in the planner generating a plan for the first sentence in Figure 1. The other sub-goal of `DESCRIBE-LIST` also causes three goals to be posted for describing each of the critical features. Since two of these are for elaborating upon the parentheses, they are not expanded because no further information is available. A skeleton of the resulting text plan is shown in Figure 3.

The system now attempts to satisfy the goal `DESCRIBE-DATA-ELEMENTS` by finding an appropriate plan. Data elements can be of three types: numbers, symbols, or lists. The system can either communicate this information by realizing an appropriate sentence, or through examples (or both). The text type and user model constraints cause the system to pick examples. It generates two goals for the two dimensions in which the data elements can vary in: the number and the type. The goal to illustrate the number feature of data elements causes two goals to be generated:

```
GENERATE-EXAMPLE-SINGLE-ELEMENT
GENERATE-EXAMPLE-MULTIPLE-ELEMENTS
```

so as to highlight the difference in number of elements between the two examples. (Each of these goals posts further goals to actually retrieve the example and generate an appropriate prompt, etc.) The example generation algorithm ensures that the examples selected for related sub-goals (such as the two above) differ in *only* the dimension being highlighted.

The goal to illustrate the *type* dimension using examples expands into four goals: to illustrate symbols, numbers, symbols and numbers, and sub-lists as possible types of data elements. (The other combinations possible – numbers and sub-lists,

[4] We use rhetorical relations from Rhetorical Structure Theory (RST) [Mann and Thompson, 1987] to ensure the generation of coherent text – `ELABORATE` is one of the relations defined in RST that can connect parts of a text.

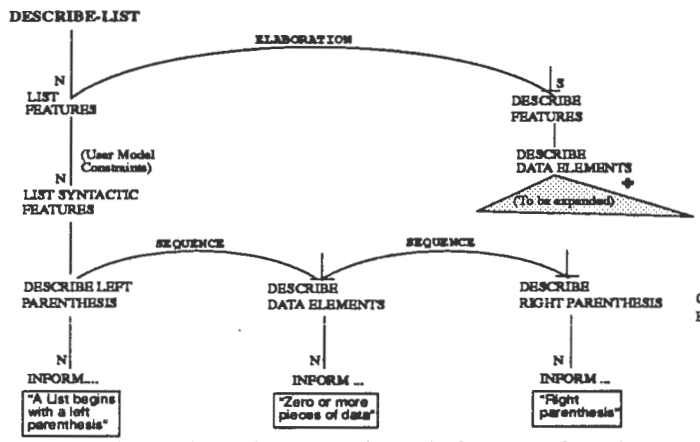


Figure 3: Plan skeleton for listing the main features of a LIST.

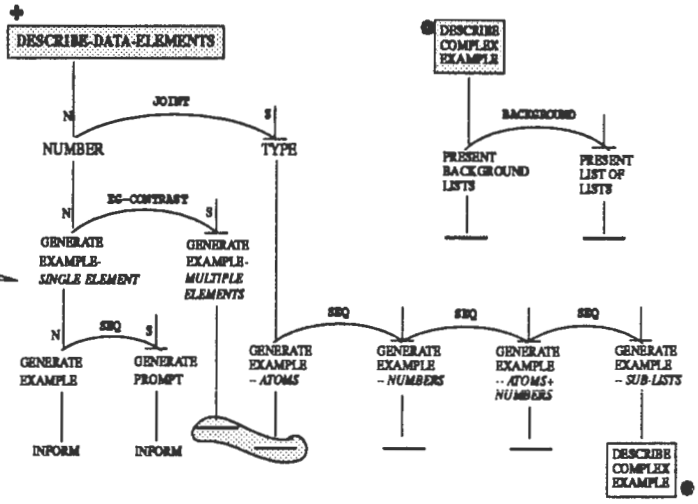


Figure 4: Partial text plan for generating the LIST examples.

symbols and sub-lists, and all three together – are also possible data element types, but are not illustrated because of a global constraint that the number of examples generated should not exceed four ([Clark, 1971]) and the system attempts to present small amounts of information at a time (because of the user model). Each of the first three goals posts appropriate goals to retrieve examples and generate prompts. The first goal of generating a LIST of atoms can be collapsed with the previously satisfied goal of generating an example of a LIST with multiple elements in our case.

In the fourth case, the user-model prevents the system from simply generating an example of a LIST which has other LISTs as its data elements. The system therefore posts two goals, one to provide background information (which presents three simple lists), and the other to build a list from these three lists. Heuristics in the system cause the system to generate text for this fourth case, rather than just a prompt. A skeleton of the second half of the complete text plan is shown in Figure 4.

The resulting discourse structure is then processed to make final decisions, such as the choice of lexical items. Finally, the completed discourse tree is passed to a a system that converts the INFORM goals into an intermediate form that is accessible to Penman, which generates the desired English output.

5 Conclusions and Future Work

This paper has largely focused on the issues that need to be addressed for an effective presentation of information in the form of a description with accompanying examples. We have identified and outlined the various questions that need to be considered, and shown through the use of examples in the domain of LISP, how some of these may be computationally implemented. We have integrated a text generator with an example generator, and have shown how this framework can be used in the generation of coherent and effective descriptions. Our work is based on an analysis of actual instructional materials and books and other studies on examples. It illustrates the possibility of planning and generating examples to illustrate definitions and explanations, by considering both of them together during the planning operation. This method also takes into account various linguistic theories on the order of presentation of facts in the descriptions, and extends them to the presentation of accompanying examples. Our system recognizes the importance of information that is usually implicit in the sequence order, and maintains information in the discourse tree that would allow it to generate prompting information. We have illustrated our framework with a brief trace of an actual description that uses examples. Our work expands on previous work that has been limited to the inclusion of examples with text – without explicit regard for many of these factors such as sequence, content of each example and prompts.

In future work, we shall investigate questions on issues such as *when* an example should be generated, and how a previous one may be easily re-used after appropriate modification.

Acknowledgments

We wish to express our gratitude and appreciation to Professors Edwina Rissland and Beverly Woolf for providing us with many references and pointers to related work.

References

[Bruner, 1966] Jerome S. Bruner. *Toward a Theory of Instruction*. Oxford University Press, London, U.K., 1966.

[Carnine, 1980] Douglas W. Carnine. Two Letter Discrimination Sequences: High-Confusion-Alternatives first versus Low-Confusion-Alternatives first. *Journal of Reading Behaviour*, XII(1):41–47, Spring 1980.

[Charney *et al.*, 1988] Davida H. Charney, Lynne M. Reder, and Gail W. Wells. Studies of Elaboration in Instructional Texts. In Stephen Doheny-Farina, editor, *Effective Documentation: What we have learned from Research*, chapter 3, pages 48–72. The MIT Press, Cambridge, MA., 1988.

This shows the so called 48combined effect, etc. Check this out.

[Clark, 1971] D. C. Clark. Teaching Concepts in the Classroom: A Set of Prescriptions derived from Experimental Research. *Journal of Educational Psychology Monograph*, 62:253–278, 1971.

[Engelmann and Carnine, 1982] Siegfried Engelmann and Douglas Carnine. *Theory of Instruction: Principles and Applications*. Irvington Publishers, Inc., New York, 1982.

[Feldman and Klausmeier, 1974] Katherine Voerwerk Feldman and Herbert J. Klausmeier. The effects of two kinds of definitions on the concept attainment of fourth- and eighth-grade students. *Journal of Educational Research*, 67(5):219–223, January 1974.

[Feldman, 1972] Katherine Voerwerk Feldman. The effects of the number of positive and negative instances, concept definitions, and emphasis of relevant attributes on the attainment of mathematical concepts. In *Proceedings of the Annual Meeting of the American Educational Research Association*, Chicago, Illinois, 1972.

[Gillingham, 1988] Mark G. Gillingham. Text in Computer-Based Instruction: What the Research Says. *Journal of Computer-Based Instruction*, 15(1):1–6, Winter 1988.

[Giora, 1988] Rachel Giora. On the informativeness requirement. *Journal of Pragmatics*, 12:547–565, 1988.

[Houtz *et al.*, 1973] John C. Houtz, J. William Moore, and J. Kent Davis. Effects of Different Types of Positive and Negative Examples in Learning "non-dimensioned" Concepts. *Journal of Educational Psychology*, 64(2):206–211, 1973.

[Klausmeier and Feldman, 1975] Herbert J. Klausmeier and Katherine Voerwerk Feldman. Effects of a Definition and a Varying Number of Examples and Non-Examples on Concept Attainment. *Journal of Educational Psychology*, 67(2):174–178, 1975.

[Klausmeier *et al.*, 1974] Herbert J. Klausmeier, E. S. Ghatala, and D. A. Frayer. *Conceptual Learning and Development, a Cognitive View*. Academic Press, New York, 1974.

[Klausmeier, 1976] Herbert J. Klausmeier. Instructional Design and the Teaching of Concepts. In J. R. Levin and V. L. Allen, editors, *Cognitive Learning in Children*. Academic Press, New York, 1976.

[Litchfield *et al.*, 1990] Brenda C. Litchfield, Marcy P. Driscoll, and John V. Dempsey. Presentation Sequence and Example Difficulty: Their Effect on Concept and Rule Learning in Computer-Based Instruction. *Journal of Computer-Based Instruction*, 17(1):35–40, Winter 1990.

[MacLachlan, 1986] James MacLachlan. Psychologically Based Techniques for Improving Learning within Computerized Tutorials. *Journal of Computer-Based Instruction*, 13(3):65–70, Summer 1986.

[Mann and Thompson, 1987] William Mann and Sandra Thompson. Rhetorical structure theory: a theory of text organization. In Livia Polanyi, editor, *The Structure of Discourse*. Ablex Publishing Corporation, Norwood, New Jersey, 1987. Also available as USC/Information Sciences Institute Technical Report Number RS-87-190.

[Mann, 1983] William C. Mann. An overview of the Penman text generation system. Technical Report ISI/RR-83-114, USC/Information Sciences Institute, 1983.

[Markle and Tiemann, 1969] S. M. Markle and P. W. Tiemann. *Really Understanding Concepts*. Stipes Press, Urbana, Illinois, 1969.

[Merrill and Tennyson, 1978] M. David Merrill and Robert D. Tennyson. Concept Classification and Classification Errors as a function of Relationships between Examples and Non-Examples. *Improving Human Performance Quarterly*, 7(4):351–364, Winter 1978.

[Moore and Paris, 1989] Johanna D. Moore and Cécile L. Paris. Planning text for advisory dialogues. In *Proceedings of the Twenty-Seventh Annual Meeting of the Association for Computational Linguistics*, pages 203 – 211, Vancouver, British Columbia, June 1989.

[Moore and Paris, 1991] Johanna D. Moore and Cécile L. Paris. Discourse Structure for Explanatory Dialogues. Presented at the Fall AAAI Symposium on Discourse Structure in Natural Language Understanding and Generation, November 1991.

[Moore and Paris, 1992] Johanna D. Moore and Cécile L. Paris. User models and dialogue: An integrated approach to producing effective explanations. To appear in the 'User Model and User Adapted Interaction Journal', 1992.

[Moore, 1989a] Johanna D. Moore. *A Reactive Approach to Explanation in Expert and Advice-Giving Systems*. PhD thesis, University of California, Los Angeles, 1989.

[Moore, 1989b] Johanna Doris Moore. *A Reactive Approach to Explanation in Expert and Advice-Giving Systems*. PhD thesis, University of California – Los Angeles, 1989.

[Paris, 1987] Cécile L. Paris. *The Use of Explicit User Models in Text Generation*. PhD thesis, Columbia University, October 1987.

[Paris, 1991a] Cécile L. Paris. Generation and Explanation: Building an explanation facility for the Explainable Expert Systems framework. In C. Paris, W. Swartout, and W. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 49 – 81. Kluwer Academic Publishers, Boston/Dordrecht/London, 1991.

[Paris, 1991b] Cécile L. Paris. The role of the user's domain knowledge in generation. *Computational Intelligence*, 7 (2):71 – 93, May 1991.

[Pirolli, 1991] Peter Pirolli. Effects of Examples and Their Explanations in a Lesson on Recursion: A Production System Analysis. *Cognition and Instruction*, 8(3):207–259, 1991.

[Reder *et al.*, 1986] Lynne M. Reder, Davida H. Charney, and Kim I. Morgan. The Role of Elaborations in learning a skill from an Instructional Text. *Memory and Cognition*, 14(1):64–78, 1986.

[Reiser *et al.*, 1985] Brian J. Reiser, John R. Anderson, and Robert G. Farrell. Dynamic Student Modelling in an Intelligent Tutor for Lisp Programming. In *Proceedings of the*

Ninth International Conference on Artificial Intelligence, pages 8–14. IJCAI-85 (Los Angeles), 1985.

[Rissland and Ashley, 1986] Edwina L. Rissland and Kevin D. Ashley. Hypotheticals as Heuristic Device. In *Proceedings of the National Conference on Artificial Intelligence*, pages 289–297. AAAI, 1986.

[Rissland *et al.*, 1984] Edwina L. Rissland, Eduardo M. Valcarce, and Kevin D. Ashley. Explaining and Arguing with Examples. In *Proceedings of the National Conference on Artificial Intelligence*, pages 288–294. AAAI, August 1984.

[Rissland, 1981] Edwina L. Rissland. Constrained Example Generation. COINS Technical Report 81-24, Department of Computer and Information Science, University of Massachusetts, Amherst, MA., 1981.

[Rissland, 1983] Edwina L. Rissland. Examples in Legal Reasoning: Legal Hypotheticals. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 90–93, Karlsruhe, Germany, 1983. IJCAI.

[Suthers and Rissland, 1988] Daniel D. Suthers and Edwina L. Rissland. Constraint Manipulation for Example Generation. COINS Technical Report 88-71, Computer and Information Science, University of Massachusetts, Amherst, MA., 1988.

[Tennyson and Park, 1980] Robert D. Tennyson and Ok-Choon Park. The Teaching of Concepts: A Review of Instructional Design Research Literature. *Review of Educational Research*, 50(1):55–70, Spring 1980.

[Tennyson and Tennyson, 1975] Robert D. Tennyson and C. L. Tennyson. Rule Acquisition Design Strategy Variables: Degree of Instance Divergence, Sequence and Instance Analysis. *Journal of Educational Psychology*, 67:852–859, 1975.

[Tennyson *et al.*, 1975] Robert D. Tennyson, M. Steve, and R. Boutwell. Instance Sequence and Analysis of Instance Attribute Representation in Concept Acquisition. *Journal of Educational Psychology*, 67:821–827, 1975.

[Touretzky, 1984] David S. Touretzky. *LISP: A Gentle Introduction to Symbolic Computation*. Harper & Row Publishers, New York, 1984.

[Ward and Sweller, 1990] Mark Ward and John Sweller. Structuring Effective Worked Examples. *Cognition and Instruction*, 7(1):1 – 39, 1990.

[Werth, 1984] Paul Werth. *Focus, Coherence and Emphasis*. Croom Helm, London, England, 1984.

[Woolf and McDonald, 1984] Beverly Woolf and David D. McDonald. Context-Dependent Transitions in Tutoring Discourse. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 355–361. AAAI, 1984.

[Woolf and Murray, 1987] Beverly Woolf and Tom Murray. A Framework for Representing Tutorial Discourse. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 189–192. IJCAI, 1987.

[Woolf *et al.*, 1988] Beverly Park Woolf, Daniel Suthers, and Tom Murray. Discourse Control for Tutoring: Case Studies in Example Generation. COINS Technical Report 88-49, Computer and Information Science, University of Massachusetts, 1988.

Formalizing Plan Justifications

Eugene Fink *
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L3G1
efink@violet.waterloo.edu

Qiang Yang *
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L3G1
qyang@logos.waterloo.edu

Abstract

This paper formalizes the notion of *justified plans*, which captures the intuition behind "good" plans. A justified plan is one that does not contain operators which are not necessary for achieving a goal. The importance of formalizing this notion is due to two reasons. First, it gives rise to methods for optimizing a given plan by removing "useless" operators. Second, several important concepts describing abstraction hierarchies are defined via justified plans. In the past, relatively few attempts have been made to formalize such a notion. This paper defines several different kinds of plan justifications, presents algorithms for finding a justified version of a plan, and shows that the task of finding the *best possible* justified version of a plan is NP-complete. Finally, it presents a greedy algorithm for finding a near-optimal justified plan in polynomial time.

1 Introduction

While searching for a plan that achieves a certain goal, we wish to find an efficient plan, which does not contain "useless" steps. Such a plan can be obtained from an inefficient plan by removing all operators that are not necessary for achieving the goal. For example, suppose that one wishes to prepare tea, by following the plan: "put a tea bag into a cup; boil water in a kettle; pour water into the cup". Suppose that later on one discovers that the kettle already contains hot water. Then the second step of the plan, "boil water", is no longer necessary for achieving the goal. After removing the second step, the resulting plan "put a tea bag into a cup; pour water into the cup" contains fewer steps while still achieving the same goal. The operation of removing useless operators from a plan is known as *justification*. The main purpose of our paper is to formalize different ways of performing plan justifications.

One application of plan justification is to augment a non-optimal planner such as STRIPS with an optimization routine. The resulting plan will then be more efficient to execute. Another application is reusing old plans. Suppose that we have found a plan for achieving goals G_1, G_2, and G_3. Later on we may use the same plan to achieve the goal G_1 alone. In this case we wish to find a subset of the initial plan which is "relevant" to achieving G_1, by removing all unnecessary operators. Thus, justification is useful for adapting an old plan to new situations.

The notion of justified plans is important not only for the purpose of optimizing plans, but also for abstract problem solving. Several important concepts describing the algorithms for generating abstraction hierarchies are defined via justified plans. For example, the theoretical concepts underlying Knoblock's planner ALPINE [Knoblock, 1990] are based on the notions of *justified plans*. Other results that depend on this notion are presented in [Tenenberg and Yang, 1990], [Knoblock *et al.*, 1991], and [Bacchus and Yang, 1991].

In spite of the importance of the concept of justified plans, relatively few efforts have been made to explore different kinds of justification. This paper begins to address this problem by formalizing and extending the previous work. We first consider the notion of *backward justified* plans that researchers have used before, which guarantees that each operator in a plan establishes a literal necessary for achieving a goal. We then present a definition of *well-justified* plans. Informally, a plan is well-justified if none of its operators may be omitted without violating the correctness of the plan. We also compare well-justified and backward justified plans in terms of their qualities. Finally, we consider the task of finding the "best possible" justification of a given plan, a subplan of a given plan that cannot be further optimized by removing any subset of its operators. We show that the task of finding such a subplan is NP-complete. To satisfy the practical need for efficient planning, we present a greedy algorithm that finds a near-optimal justification in polynomial time.

We begin by presenting a formal description of the problem space language used for describing our results. Then we consider each type of justification in turn.

*The authors are supported in part by a scholarship and grants from the Natural Sciences and Engineering Research Council of Canada.

2 Problem Space Language

A *planning domain* consists of a set of literals $\mathcal{L}$ and a set of operators O. Each *operator* α is defined by a set of *precondition literals* $Pre(\alpha)$ and *effect literals* $Eff(\alpha)$.

A *state* of the world is a set of literals. Applying an operator α to some state produces a new state, where all literals from $Eff(\alpha)$ hold, and all literals that do not conflict with $Eff(\alpha)$ are left unchanged. For example, suppose p_1 and p_2 are some atomic statements in a problem domain. The corresponding literals are p_1, p_2, $\neg p_1$, and $\neg p_2$. Let $Eff(\alpha) = \{\neg p_1\}$. Then applying α to the state $S = \{p_1, p_2\}$ produces the new state $S' = \{\neg p_1, p_2\}$.

A *linearly ordered plan* $\overline{\Pi} = (\alpha_1, \ldots, \alpha_n)$ is a sequence of operators, which can be applied to some *initial state* by executing each operator in order. A plan $\overline{\Pi} = (\alpha_1, \ldots, \alpha_n)$ is *legal* relative to an initial state S_0 if the preconditions of each operator are satisfied in the state in which the operator is applied, i.e. $\forall i \in [1 \ldots n]$, $Pre(\alpha_i) \subseteq S_{i-1}$. A plan $\overline{\Pi}$ *solves* a *goal state* S_g if $\overline{\Pi}$ is legal and the goal is satisfied in the final state: $S_g \subseteq S_n$. A legal plan that solves the goal S_g is called *correct* relative to S_g.

A *partially ordered plan* is a set of operators $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ with a partial order $\prec_\Pi$ on it. This partial order represents the time-precedence relation between operators: $\alpha_1 \prec_\Pi \alpha_2$ means that α_1 must be executed before α_2. A linearly ordered plan $\overline{\Pi}$ is a *linearization* of Π if it contains all the operators of Π and the order defined by $\prec_\Pi$ is not violated, that is for any α_i and α_j, if $\alpha_i \prec_\Pi \alpha_j$, then α_i occurs before α_j in $\overline{\Pi}$. A partially ordered plan is legal if all its linearizations are legal, and it solves a goal S_g if all its linearizations do. Throughout the remainder of the paper all plans are partially ordered unless otherwise specified.

A plan Π' is called a *subplan* of Π if it is obtained from Π by eliminating one or more operators. The precedence relation between the remaining operators must be preserved. That is, Π' is a subplan for Π if and only if

$\forall \alpha_1, \alpha_2 \in \Pi'$
(1) $\alpha_1, \alpha_2 \in \Pi$ and
(2) $\alpha_1 \prec_{\Pi'} \alpha_2 \Leftrightarrow \alpha_1 \prec_\Pi \alpha_2$.

3 Backward Justification

To formalize the notion of justified plans, we first generalize the concept of establishment defined in [Knoblock *et al.*, 1991] to partially ordered plans.

Definition 1 (Establishment) *Let $\overline{\Pi}$ be a legal linearly ordered plan. Let α_1 and α_2 be two operators of the plan $\overline{\Pi}$, $\alpha_1, \alpha_2 \in \overline{\Pi}$, $l \in \mathrm{Eff}(\alpha_1)$, and $l \in \mathrm{Pre}(\alpha_2)$. Then α_1* establishes *l for α_2 if*

1. $\alpha_1 \prec \alpha_2$, and
2. $\forall \alpha \in \overline{\Pi}$, if $\alpha_1 \prec \alpha \prec \alpha_2$ then $l, \neg l \notin \mathrm{Eff}(\alpha)$

We say that α_1 possibly establishes *a literal l for α_2 in a partially ordered plan Π if it establishes l for α_2 in at least one linearization of Π.*

Intuitively this means that the precondition l of the operator α_2 holds before the execution of α_2, and α_1 is the last operator that achieves it.

Definition 2 (Backward justification) *Let Π be a legal plan that achieves a goal S_g. An operator $\alpha \in \overline{\Pi}$ is called* backward justified *if $\exists l \in \mathrm{Eff}(\alpha)$ such that α possibly establishes l either for the goal S_g or for another backward justified operator.*

We say that a plan Π is backward justified, if all its operators are backward justified. This definition of justification was used in the planner ALPINE [Knoblock *et al.*, 1991]. For linearly ordered plans it is equivalent to the definition stated in [Tenenberg and Yang, 1990]. For partially ordered plans, backward justification is weaker then the justification described in [Tenenberg and Yang, 1990].

An operator is backward justified if it possibly establishes some literal necessary for achieving the goal. However, it may happen that l has already been established before α, and then α is useless in Π. Thus backward justified operators are not "truly justified". We illustrate this point with the following example.

Assume that one has a kettle with hot water and an empty cup, and wishes to have a cup of hot water. The following plan achieves the goal

1. Pour water into the cup.
2. Put the cup into a microwave.

The second operator is backward-justified, because it makes the water hot, while no other operator *after* it achieves the same goal. However, this operator may still be removed, because the water was already hot before its execution. Thus, the second operator is not truly justified.

Observe that if α is the last operator in some linearization of a plan that does not establish any goal literals or operator preconditions, then it can be removed without violating correctness of the plan. After its removal, the plan remains correct. We could then apply the same procedure recursively, until no more operators can be removed without violating the correctness of the plan. This is the basis of the algorithm for finding a backward justified plan.

The algorithm is shown in Table 3a. It first linearizes the plan Π. Then it checks whether or not the last operator α in the plan establishes a goal. If α doesn't establish any goal, then it should be removed. Then the algorithm considers the rest of the operators, going from the end to the beginning of the plan. Each operator that does not establish any literal for the goal nor for any other operator is removed. Observe that when we consider an operator, all operators after this operator that are not backward justified are already removed. Thus, the operator is not removed only if it establishes a literal for some *backward justified* operator, which means that the operator itself is backward justified. Since the algorithm proceeds from the end to the beginning of the plan, the resultant plan is called *backward* justified.

To check the condition in line 5, we need to check for every $\alpha_1 \in \Pi$ with the precondition l, if there is a linearization of the plan Π where no operator between α and α_1 establishes or removes the literal l. In other words, for each operator that achieves l or $\neg l$, we have

to check whether it is necessarily between α and α_1[1]. If there is no such an operator, α possibly establishes l for α_1. If the order of operators is represented by a transitively closed graph, this condition may be checked in $O(|\Pi|)$ time for each α_1, where $|\Pi|$ is the number of operators in the initial plan. Therefore the search of α_1 established by α takes $O(|\Pi|^2)$ time. The overall running time of the algorithm is $O(E \cdot |\Pi|^2)$, where $E = \sum_{\alpha \in O} |Eff(\alpha)|$, and $|Eff(\alpha)|$ is the number of literals in the set of effects of α.

4 Well Justification

Definition 3 (Well-justification) *An operator α_i in a linearly ordered plan $\overline{\Pi}$ is called* well-justified *if $\exists l \in$ Eff(α_i) such that α_i establishes l for some operator or for the goal S_g, and l does not hold before α_i, that is $l \notin S_{i-1}$.*

An operator in a partially ordered plan is called well-justified if it is well-justified in at least one linearization of the plan.

A plan is well-justified if all its operators are well-justified. Intuitively, an operator is well-justified if it establishes some literal which has not been established before, and which is necessary for executing some other operator. This means that if we remove a well-justified operator from a plan, the plan is no longer correct. We state this result as a lemma.

Lemma 1 *An operator is well-justified if and only if we cannot remove it from the plan without violating correctness of the plan.*

The next theorem follows directly from the lemma.

Theorem 1 *A plan is well-justified if and only if there is no operator that can be removed without violating correctness of the plan.*

This theorem shows that well-justification captures the intuition behind "good" plans: a well-justified plan does not contain any operator that is not necessary for achieving the goal. Recall that if a plan Π is not backward justified, then any operator that is not backward justified may be removed without violating the correctness of Π. By Theorem 1 this means that Π is not well-justified either. Thus, every well-justified plan is backward justified. In other words, well-justification is stronger than backward justification.

For a given legal plan, there might be several distinct well-justified subplans of the same plan, as the following example demonstrates.

Suppose one has a kettle of cold water, and needs a cup of hot water. The following plan would lead to the desired result

1. Boil water by putting the kettle onto a stove.
2. Pour the water into the cup.
3. Put the cup into a microwave.

This plan is not well-justified, because either the first or third operator may be skipped without violating the correctness of the plan. Thus, the plan has two well-justified subplans: one of them consists of the first two operators, and the other consists of the last two.

The simple algorithm that finds a well-justified subplan of a given plan is shown in Table 3b. The running time of the algorithm that checks correctness of a given plan is $O(P \cdot |\Pi|^2)$, where $P = \sum_{\alpha \in O} |Pre(\alpha)|$, and $|Pre(\alpha)|$ is the number of literals in the set of preconditions of α. Therefore the overall running time of the algorithm is $O(P \cdot |\Pi|^4)$.

5 Perfect Justification

While well-justified plans cannot contain unnecessary operators, they still may contain unnecessary *groups of operators*. This means that while no single operator may be eliminated from the plan, several operators may be eliminated *together*. In particular, a linearly ordered well-justified plan $\overline{\Pi} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ may contain a *cycle*, which means that the same state is achieved twice during the plan execution. Formally, a sequence of operators $\alpha_{i+1}, \alpha_{i+2}, \ldots, \alpha_j$ in $\overline{\Pi}$ is called a cycle if $S_i \supseteq S_j$. Observe that we may eliminate a cycle from $\overline{\Pi}$ without violating correctness of $\overline{\Pi}$. For example, consider the following plan of boiling water:

1. Fill a cup with water.
2. Empty the cup.
3. Fill the cup with water again.
4. Put the cup into a microwave.

This plan is well-justified: we cannot skip *operator 2*, because then we could not fill the cup again; and we cannot skip *operator 3*, because the cup has to be full when we put it into a microwave. However, we may skip *operators 2* and *3* together. To formalize this observation, we introduce the notion of perfect justification.

Intuitively, a plan is perfectly justified if no subset of its operators may be removed from the plan. In other words, this is the "best possible" justification.

Definition 4 (Perfect justification) *A correct plan Π is called* perfectly justified *w.r.t. a goal S_g if it does not have any legal proper subplan that achieves the goal.*

Just by definition perfect justification is stronger than all justifications discussed above. Unfortunately, a perfect justification of a given plan cannot be found in polynomial time. In this paper we show that the task to find a perfect justification of a given plan is NP-hard, even for linearly ordered plans. Moreover, it is NP-hard to check whether a linearly ordered plan is perfectly justified.

Theorem 2 *Suppose we are given a linearly ordered plan $\overline{\Pi}$ with an initial state S_0 and a goal S_g, and we wish to determine whether this plan is perfectly justified. This problem is NP-complete.*

Sketch of the proof. The problem is trivially NP, since, given a subplan of $\overline{\Pi}$, we may check whether this subplan is legal and achieves the goal in polynomial time. To show that the problem is NP-hard, we reduce 3-clause satisfiability problem to our problem.

Suppose we are given a 3-clause conjunctive normal form with n distinct variables $V_1, V_2, \ldots, V_n$, and k dis-

[1] An operator β is *necessarily* between α and α_1, if $\alpha \prec \beta$ and $\beta \prec \alpha_1$.

operators	preconds	effects
α_i (for each $i \in [1 \ldots n]$)	—	$v_i^+, \neg v_i^-$
β_{ij} (for each $V_i \in C_j$)	v_i^+	c_j, p_{ij}
γ_{ij} (for each $\neg V_i \in C_j$)	v_i^-	c_j, p_{ij}
δ	$\neg v_1^-, \neg v_2^-, \ldots, \neg v_n^-$	$v_1^-, v_2^-, \ldots, v_n^-$ and all $\neg p_{ij}$'s

Table 1: Operators in the proof of NP-completeness

tinct clauses $C_1, C_2, \ldots, C_k$. For each variable V_i we introduce two predicates, v_i^+ and v_i^-. For each clause C_j we introduce a corresponding predicate c_j. Finally, for each pair (V_i, C_j), where V_i is a variable *in* the clause C_j, we introduce a predicate p_{ij}. We define a problem domain that contains all literals defined by the introduced predicates. (Each predicate p gives rise to the two literals: p and $\neg p$.) We define operators in our problem domain as shown in Table 1.

We define an initial state S_0 as follows

- $(\forall i \in [1 \ldots n])$ $v_i^- = True$ and $v_i^+ = False$
- $(\forall j \in [1 \ldots k])$ $c_j = False$
- for all predicates p_{ij} in our domain, $p_{ij} = True$

and a goal S_g as follows:

- $(\forall j \in [1 \ldots k])$ $c_j = True$
- for all predicates p_{ij} in our domain, $p_{ij} = True$

Now we present a linearly ordered plan with the initial state S_0:

$$\overline{\Pi}=(\alpha_1, \alpha_2, \ldots, \alpha_n, \delta, \text{ all } \beta_{ij}\text{'s, all } \gamma_{ij}\text{'s})$$

where the order of β_{ij}'s and γ_{ij}'s is arbitrary. It is straightforward to verify that the plan $\overline{\Pi}$ is legal and solves the goal S_g. Further, one may show that *if* $\overline{\Pi}$ has a legal proper subplan that achieves the goal, this subplan cannot contain δ, for if some of α-operators is removed from $\overline{\Pi}$, the preconditions of δ are not satisfied, and if one or more β's or γ's are removed, δ interferes with achieving the goal. Thus, if Π has a legal proper subplan, this subplan must have the form

$$\overline{\Pi}'=(\alpha_{k_1}, \alpha_{k_2}, \ldots, \alpha_{k_m}, \text{ some } \beta_{ij}\text{'s, some } \gamma_{ij}\text{'s})$$

It may be shown that the following two statements are equivalent:

- $\overline{\Pi}$ has a legal subplan of the form $\overline{\Pi}'$ that achieves the goal
- $V_{k_1} = V_{k_2} = \ldots = V_{k_m} = True$, and all other variables $V_{k_{m+1}} = \ldots = V_{k_n} = False$ is a satisfying assignment of the conjunctive normal form

Thus, the conjunctive normal form has a satisfying assignment *if and only if* the plan $\overline{\Pi}$ has a legal proper subplan that achieves the goal, in other words, if and only if $\overline{\Pi}$ is not perfectly justified. □

Corollary 1 *The problem to find a perfectly justified subplan of a given plan is NP-complete.*

6 Greedy justification

While the task of finding the best possible justified plan is NP-hard, one can design a greedy algorithm that finds an "almost" perfect justification. To check "usefulness" of some operator α in a plan Π, the algorithm proceeds as follows. First, it removes an operator α from the plan. After α has been removed, some operators of Π may become *illegal*, which means that now their preconditions are *not* satisfied before their execution. The algorithm removes every operator which is the first illegal operator in at least one linearization of Π. Then the algorithm examines the resulting plan, finds the remaining illegal operators, and again removes all earliest illegal operators. The algorithm repeats this step until the plan becomes legal. If this plan still solves the goal, then the initially removed operator α was not useful, and we say that α is not *greedily justified.*

The description of the algorithm is presented in Table 3c. The set of illegal operators in line 3 may be found in $O(P \cdot |\Pi|^2)$ time. The same time is required for correctness checking in line 6. Finally, computing the set *Earliest_Illegals* in line 4 requires $O(|\Pi|^2)$ time, if the order of operators is represented by a transitively closed graph. The overall running time of the algorithm is $O(P \cdot |\Pi|^3)$.

As an example, consider again the water-boiling plan:

1. Fill a cup with water.
2. Empty the cup.
3. Fill the cup with water again.
4. Put the cup into a microwave.

Suppose we remove *operator 2.* Now *operator 3* is illegal, because we cannot fill a cup which is already full, and it should be removed from the plan. The resulting plan is:

1. Fill a cup with water.
4. Put the cup into a microwave.

which is legal and solves the goal. Thus, *operator 2* in the initial plan is not greedily justified.

If an operator α in a plan is not well-justified, and we use the algorithm *Greedy_Justify_Checking* to check the *usefulness* of α, then α will be removed at the first step of execution. The resultant plan is legal and solves the goal. Thus, if an operator is not well-justified, it is not greedily justified either, and therefore greedy justification is stronger than well-justification. Also, the algorithm is able to detect and remove cycles: if a linearly ordered plan contains a cycle $\alpha_{i+1}, \alpha_{i+2}, \ldots, \alpha_j$, then, while testing usefulness of α_{i+1}, the algorithm will remove α_{i+1}, then α_{i+2}, then α_{i+3}, and so on till α_j, and then it receives a legal subplan that solves the goal.

A plan is greedily justified if all its operators are greedily justified. It follows from the above discussion that such a plan is always well-justified and does not contain cycles. An algorithm that finds a greedily justified subplan of a plan Π may be briefly described as follows

kind of subplan	running time	
perfectly justified	NP-complete	stronger justification
greedily justified	$O(P \cdot \|\Pi\|^5)$	$\uparrow$
well-justified	$O(P \cdot \|\Pi\|^4)$	$\downarrow$
backward justified	$O(E \cdot \|\Pi\|^2)$	weaker justification

Table 2: Kinds of justified subplans and running time to find them

1. for each operator of the plan Π
1a. use *Greedy_Justify_Checking* to check if the operator is greedily justified
1b. if it is *not* greedily justified, we receive some correct subplan Π' of Π; then we recursively call the algorithm for Π', to find its greedy justification
2. if all operators are greedily justified, then our plan is greedily justified, and so a greedily justified subplan of the initial plan is found

It may be shown that *Greedy_Justify_Checking* is called at most $|\Pi|^2$ times, and thus the running time of the algorithm is $O(P \cdot |\Pi|^5)$.

The running time may be considerably improved in the case of a linearly ordered plan. The algorithm for this case is shown in Table 3d. To determine whether some operator α is greedily justified, the algorithm removes this operator and executes the remaining operators in order. If an illegal operator is encountered, the algorithm removes this operator and continues to execute the plan. Thus, it removes all illegal operators and receives the final state that the plan achieves with the illegal operators removed. If the goal is not achieved, then the initially removed operator α is greedily justified. On the other hand, if the new plan achieves the goal, than it is an optimized version of the initial plan. Then we apply our algorithm recursively to check if this new, shorter plan is greedily justified. The running time of the algorithm is $O((P+E)|\overline{\Pi}|^2)$, providing the problem domain contains the finite number of literal classes.

7 Conclusion and Open Problems

This paper formalizes the intuition behind "good" partially and linearly ordered plans. Table 2 presents different kinds of justification and running time necessary to find justified subplan of a plan for each kind of justification. Running time is presented for algorithms dealing with partially ordered sets. Recall that the algorithm to find a greedily justified version of a linearly ordered plan is much faster; it takes only $O((P+E) \cdot |\overline{\Pi}|^2)$ time.

The table may be viewed as a spectrum of justified plans. On one end of the spectrum plans are backward justified. A backward justified subplan of a given plan is not hard to find, but it may contain some "useless" operators. The other end of the spectrum contains perfectly justified plans. They cannot have any useless operators, but it is NP-hard to find a perfectly justified subplan of a given plan.

The results of this paper may be used for creating abstraction hierarchies. According to the definition of *ordered abstraction hierarchies* presented in [Knoblock *et al.*, 1991], different kinds of justification give rise to different ordered hierarchies. *More* restrictive kinds of justifications give rise to *less* restrictive conditions for building an abstraction hierarchy, resulting in finer-grained hierarchies. So, using the definitions of well-justified and greedily justified plans, we may build finer ordered abstraction hierarchies than those generated by Knoblock's ALPINE. The theoretical results and algorithms that allow us to build such finer hierarchies are presented in [Fink, 1992].

References

[Bacchus and Yang, 1991] Fahiem Bacchus and Qiang Yang. The downward refinement property. In *Proceedings of the 12th IJCAI*, pages 286–292, Sydney, Australia, August 1991.

[Fink, 1992] Eugene Fink. Justified plans and ordered hierarchies. Master's thesis, University of Waterloo, Department of Computer Science, Waterloo, Ont., Canada, Forthcoming 1992.

[Knoblock, 1990] Craig A. Knoblock. Learning abstraction hierarchies for problem solving. In *Proceedings of Eighth National Conference on Artificial Intelligence*, pages 923–928, Boston, MA, 1990.

[Knoblock *et al.*, 1991] Craig Knoblock, Josh Tenenberg, and Qiang Yang. Characterizing abstraction hierarchies for planning. In *Proceedings of the 9th AAAI*, Anaheim, CA, 1991.

[Knoblock, 1991] Craig A. Knoblock. *Automatically Generating Abstractions for Problem Solving.* PhD thesis, School of Computer Science, Carnegie Mellon University, 1991. Tech. Report CMU-CS-91-120.

[Sacerdoti, 1974] Earl Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[Tenenberg, 1988] Josh Tenenberg. *Abstraction in Planning.* PhD thesis, University of Rochester, Dept. of Computer Science, Rochester, NY, May 1988.

[Tenenberg and Yang, 1990] Josh Tenenberg and Qiang Yang. ABTWEAK: abstracting a nonlinear, least commitment planner. In *Proceedings of Eighth National Conference on Artificial Intelligence*, pages 923–928, Boston, MA, 1990.

Table 3: ALGORIHMS

Backward_Justification
1. let $\overline{\Pi}$ be some linearization of Π;
2. **for** α := (last operator of $\overline{\Pi}$) **downto** (first operator of $\overline{\Pi}$) **do**
   begin
3. *Justified* := *False*;
4. **for each** $l \in Eff(\alpha)$ **do**
5. **if** ($\exists \alpha_1 \in \Pi$ sth α establishes l for α_1) **or** (α establishes l for S_g)
6. **then** /* α is backward justified */ *Justified* := *True*;
7. **if** *Justified=False* /* α is not backward justified */
8. **then** remove α from the plan Π;
   end

(a) Finding the backward justified subplan of a given plan.

Well_Justification
1. **repeat**
2. **for each** $\alpha \in \Pi$ **do**
3. **if** Π without α is legal and achieves the goal
4. **then** remove α from Π
5. **until** no operator is removed during the last execution of the loop

(b) Finding a well-justified subplan of a given plan.

Greedy_Justify_Checking(Π,α)
1. remove α from Π
2. **repeat**
3. *Illegals* := "the set of illegal operators of Π";
4. *Earliest_Illegals* := $\{\alpha' \in Illegals \mid (\forall \alpha_1 \in \Pi)\ \alpha_1 \prec \alpha' \Rightarrow \alpha_1 \notin Illegals\}$
   /* That is *Earliest_Illegals* is the set of earliest illegal operators */;
5. remove all operators of the set *Earliest_Illegals* from Π;
6. **until** Π does not contain illegal operators;
7. **if** Π still achieves the goal
8. **then return**("Π is a legal subplan of the initial plan")
9. **else return**("α in the initial plan is greedily justified")

(c) Checking if the operator α in the plan Π is greedily justified.

Linear_Well_Justification($\overline{\Pi}, S_0, S_g$)
1. **for each** $\alpha \in \overline{\Pi}$ **do**
   begin
2. $\overline{\Pi}_1$:= $\overline{\Pi}$ with α removed;
3. $S := S_0$;
4. **for** α_1:= (first operator of $\overline{\Pi}_1$) **to** (last operator of $\overline{\Pi}_1$) **do**
5. **if** $Pre(\alpha_1) \subseteq S$ /* α_1 is legal */
6. **then** /* execute α_1 */ S := state received by applying α_1 to S
7. **else** remove α_1 from $\overline{\Pi}_1$;
8. **if** $S_g \subseteq S$ /* $\overline{\Pi}_1$ achieves the goal S_g */
9. **then return**(Linear_Well_Justification($\overline{\Pi}_1, S_0, S_g$))
   end;
10. return($\overline{\Pi}$)

(d) Finding a greedily justified subplan of a linearly ordered plan.

Building Macros in Deterministic and Non-Deterministic Domains

Bertrand Pelletier
Département d'informatique
Université du Québec à Hull
C.P. 1250, succ. B
Hull, Québec
J8X 3X7
Canada
bertrand@csi.uottawa.ca

Stan Matwin
Ottawa Machine Learning Group
Department of Computer Science
University of Ottawa
Ottawa, Ontario
K1N 6N5
Canada
stan@csi.uottawa.ca

Abstract

This paper describes TWEAK$^+$, a STRIPS-like formalism for the description of operators and macros constructed by chaining operators. In TWEAK$^+$, each operator or macro is described by four lists of positive or negative literals: a precondition, a DEL list, an ADD list and a postcondition. We show how the description of macros can be efficiently computed, in certain situations, using bit vector equations (in linear time and space with respect to the number of operators contained in the macro). These situations are characterized by deterministic domains; if the domain also uses an axiomatic theory, the equivalent reduced operator descriptions must not contain disjunctions. For the other situations, the bit vector equations cannot be used, and heuristical methods to compute sufficient precondition and postcondition for macros are proposed.

1 Introduction

As [Fikes and Nilsson 1971] pointed out in their pioneering work, providing a priori a problem solver with the complete set of specialized operators for solving new problems is not realistic (although desirable). It is more appropriate that the problem solver learns new operators by "chaining together existing operators into more complex ones" [Fikes and Nilsson 1971]. Building macros also leads to more efficient problem solving by storing and reusing "building-blocks" plans. For instance, in [Pelletier 1992], macros in the form of design rules are learned and used to enhance a software tool that assists designers when constructing entity-relationship models.

This paper addresses the problem of constructing macros by chaining operators. Our focus is on the issues of efficient representation of macros (e.g., for solving more complex problems or or building higher level macros), rather than on learning macros. In other words, it is assumed that the sequence of operators is provided by some expert, or by some problem solver, and the goal is to determine its description (its precondition and its effect). In particular, it must be determined if there exists an initial condition under which the macro is executable.

To represent the basic operators and the macros created, the proposed approach uses a variation of the TWEAK formalism [Chapman 1987], a descendant of the STRIPS formalism [Fikes and Nilsson 1971].

In STRIPS, an operator is described by a precondition defining its condition of applicability (i.e., the minimal condition that must hold in the initiating state prior to executing the operator), an ADD list specifying the set of formulas that are true in the resulting state, and a DEL list specifying the set of formulas that may no longer be true in the resulting state (note that the formulas appearing in the DEL list are not necessarily false in the resulting model; their truth value is unknown). Typically, the precondition in STRIPS is an arbitrary formula, the DEL and ADD lists are restricted to a predetermined set of "allowable formulas", and no operators have an effect that is not described inside the DEL/ADD lists (the STRIPS assumption). This guarantees the soundness of the representation [Georgeff 1987].

In TWEAK, each operator is represented by a precondition (defining, as for STRIPS, its condition of applicability) and a postcondition (defining the strongest condition that is true in the state resulting from the operator application). Both the precondition and the postcondition are assumed to be finite sets of positive and/or negative literals. Any TWEAK representation can be converted into an equivalent STRIPS representation: the precondition of the operator remains the same, the positive literals of the postcondition become the elements of the ADD list, and the negative literals of the postcondition (without their negation connective) become the elements of the DEL list. As in STRIPS, the TWEAK representation assumes that the domain operators can be described without conditional actions, derived side effects, or dependencies of effects on the initiating state.

Despite the representation limitations it imposes, the TWEAK formalism is used by a number of problem solvers, particularly planners, mainly because it neatly bypasses the

frame problem [Genesereth and Nilsson 1987] (i.e., the need to specify that everything not affected by an operator remains as it was) and because it leads to efficient planning ([Chapman 1987; Kambhampati and Hendler 1990]).

Other formalisms have been proposed for representing macros. For instance, [Korf 1985] and [Tadepalli 1991] use macro tables as a compact and efficient representation to store and use macros. However, the representation requires that the domain be serially decomposable (i.e., all domain features can be ordered such that the effect of any operator on a feature value is a function of the values of that feature and all the features that precede it).

In [Fikes *et al.* 1972], macros are represented with triangle tables where the ith row corresponds essentially to the state resulting from the application in sequence of the first i operators of the macro. The operators used to build macros are represented in the STRIPS formalism; in particular, they can have arbitrary preconditions. However, elements of the triangle tables are restricted to conjunctions of literals (e.g., literals that constitute the "support" of the precondition of an operator), and so is the precondition of macros. So, macros are in fact represented via a TWEAK-like formalism. Triangle tables provide an efficient way to trace the execution of macros. However, they do not give complete description of the objects they represent: although they provide the precondition of a macro operator Op, its DEL list is not represented, and only approximations of its ADD list and its postcondition are provided (e.g., the union of cells of the last row gives a set S such that $ADD(Op) \subseteq S \subseteq postcond(Op)$).

More recently, [Prieditis 1990] describes a system that discovers iterative macro-operators. The learned operators are represented using a TWEAK-like formalism with recursive equations for the precondition, and for the DEL and ADD lists. The repeated sub-sequences are provided by the user, and the explanation-based generalization (EBG) technique [Mitchell *et al.* 1986] is used to identify and generalize their description. Then, the description of the macro-operator that repeats the sub-sequences is computed. In order to compute the description of the macro, constraints are imposed on the way the repeated sub-sequences interact (e.g., the DEL list of a sub-sequence shares common elements only with the ADD list of the very next sub-sequence).

The formalism we propose is described in Section 2, and an efficient solution is given for computing the description of macros built by chaining operators described in this formalism. In Section 3, the method is illustrated with an example from the blocks world. Section 4 proposes extensions to this formalism, and presents difficulties of deriving efficient solutions for these extensions. Section 5 presents our conclusion.

2 Computing Macros in TWEAK$^+$

In this section, we present the formalism used and derive equations for computing the description of a macro according to this formalism, given the description of the operators that the macro contains.

2.1 The TWEAK+ Formalism

The proposed formalism (called TWEAK$^+$) consists of augmenting the TWEAK representation with DEL and ADD lists, and of allowing these lists to contain negative literals.

The traditional STRIPS use of DEL and ADD lists is generalized here. In the "pure" use of DEL/ADD lists, it is not possible to specify that a given literal becomes false under the application of some operator. Indeed, states, DEL and ADD lists contain only positive literals, and the presence of a (necessarily positive) literals in the DEL list indicates only that its negation may become true. In this context, a simplified version of the closed-world assumption [Genesereth and Nilsson 1987] is used to infer the truth value of such negative literals: only the (necessarily positive) literals present in the description of the state are assumed to be true, all the other (including all negative literals) are assumed to be false.

In the formalism proposed here, negative literals are allowed for the description of the states as well as for the DEL and ADD lists, with the usual interpretation of DEL/ADD lists: the presence of a literal (negative or positive) in the DEL list indicates that its truth value is unknown in the resulting state, and the presence of a literal (negative or positive) in the ADD list indicates that its truth value is true in the resulting state. It is thus possible to indicate that a literal becomes false in the resulting state by inserting its negation inside the ADD list. Consequently, the use of the closed-world assumption becomes unnecessary.

Some other considerations could be taken into account, such as the presence of the same literal in both the DEL and ADD lists of a given operator, or the addition of a partially instantiated literal to a state. These problems are treated in [Pelletier 1992].

2.2 Bit Vector Equations

We now derive the equations for computing the description of macros. We are interested in equations involving only set operators (union, difference) because of the obvious efficiency of implementation of these operators: sets can be represented as bit vectors for efficient storage and use. These equations are thus called "bit vector equations".

We derive the equations for the propositional case. For the case where literals are predicates, two approaches can be taken: using generalized equations with a different interpretation for the union and the difference of sets, or using a two-step method (as in the construction of triangle tables in [Fikes *et al.* 1972]): (1) transfer and solve the problem into the propositional case by considering only the full instantiation of an operator once applied to a given state, and (2) generalize the solution found in (1).

The equation computing the postcondition of an operator or a macro operator Op, given its precondition precond(Op) and its lists DEL(Op) and ADD(Op), is directly derived from the way DEL and ADD lists are applied on states:

(1) $postcond(Op) = (precond(Op) - DEL(Op)) \cup ADD(Op).$

Now, we will concentrate on deriving the equations giving the precondition, the DEL list and the ADD list of a macro Op = (Op_1,...,Op_n) (the operator Op_i is executed in the state resulting in the execution of the operator Op_{i-1}). Because operators of a macro are applied in sequence, it is sufficient to consider macros of length 2. So, assume that Op = (Op_1,Op_2) is given, along with the respective description of Op_1 and Op_2: (P_1,D_1,A_1,Q_1) and (P_2,D_2,A_2,Q_2) (for the precondition, DEL list, ADD list and postcondition). The description ($P_{1,2}$,$D_{1,2}$,$A_{1,2}$,$Q_{1,2}$) of the macro Op can be obtained by following equations:

(2) $P_{1,2} = P_1 \cup (P_2 - Q_1)$
Explanation: to find the precondition of the macro, add to the precondition of Op_1 (i.e., P_1) what is required by the precondition of Op_2 but is not provided by the execution of Op_1 (i.e., $P_2 - Q_1$).

(3) $A_{1,2} = [(A_1 - D_2) \cup A_2] - P_{1,2}$
Explanation: this is derived from the order of application: $A_{1,2}$ contains everything that is added by Op_1 (except what is later on deleted by D_2), plus what is added by Op_2. From that, do not consider, as literals to add, the literals that were already true in the precondition of the strategy (i.e., $P_{1,2}$).

(4) $D_{1,2} = [[(D_1 - A_1) \cup D_2] - A_2] - \neg P_{1,2}$
(where the negation of a set is given by
(5) $\neg \{l_1, l_2, ..., l_n\} = \{\neg l_1, \neg l_2, ..., \neg l_n\}$)
Explanation: again, this is derived strictly from the order of application: $D_{1,2}$ contains everything that is deleted by Op_1 (except what is later on added by A_1), plus what is deleted by Op_2; from that do not consider what is finally added by A_2, and from that, do not consider, as literals to delete, the literals that were already false in the precondition of the strategy. (i.e., $\neg P_{1,2}$).

The equation (2) for $P_{1,2}$ can be simplified as follows:

$P_{1,2} = P_1 \cup (P_2 - Q_1)$

$= P_1 \cup (P_2 - [(P_1 - D_1) \cup A_1])$
using (1): $Q_1 = (P_1 - D_1) \cup A_1$

$= P_1 \cup (P_2 - [A_1 \cup (P_1 - D_1)])$
using $A \cup B = B \cup A$

$= P_1 \cup [(P_2 - A_1) - (P_1 - D_1)])$
using $A - (B \cup C) = (A - B) - C$

$= [P_1 \cup (P_2 - A_1)]$
$- [((P_2 - A_1) \cap (P_1 - D_1)) - P_1]$
using
$A \cup (B - C) = [A \cup B] - [(B \cap C) - A]$

(6) $= P_1 \cup (P_2 - A_1)$.
using
$((P_2 - A_1) \cap (P_1 - D_1)) - P_1 = \{\}$,
because
$(P_2 - A_1) \cap (P_1 - D_1) \subseteq (P_1 - D_1) \subseteq P_1$

The equation (4) for $D_{1,2}$ can be simplified as follows:

$D_{1,2} = [[(D_1 - A_1) \cup D_2] - A_2] - \neg P_{1,2}$

$= [(D_1 \cup D_2) - A_2] - \neg P_{1,2}$
by assumption, $D_1 \cap A_1 = \{\}$,
so $D_1 - A_1 = D_1$

(7) $= (D_1 \cup D_2) - (A_2 \cup \neg P_{1,2})$
using $(A - B) - C = A - (B \cup C)$.

Finally, the value of $Q_{1,2}$ can be determined using (1) once $P_{1,2}$, $D_{1,2}$ and $A_{1,2}$ are known.

Note that although a description can be "computed" for a macro, the macro is not necessarily executable. This is the case, for instance, when Op_1 removes literals from $P_{1,2}$ that are in P_2. To guarantee the executability of the macro, the following verifications must be made:

(a) none of $P_{1,2}$, $A_{1,2}$, $D_{1,2}$ and $Q_{1,2}$ contains both a literal and its negation
(b) $P_1 \subseteq P_{1,2}$ (this is always true, according to (6))
(c) $P_2 \subseteq (P_{1,2} - D_1) \cup A_1$
(i.e., the precondition of Op_2 is true after the execution of Op_1 under $P_{1,2}$).

3 Computing a Macro Description: an Example

This section illustrates the equations derived in the previous section with an example drawn from the blocks world ([Rich 1983], p.255), where predicates with variables have been replaced with propositions. Three primitive operators are presented:

Op_1 = pick_up_A,
Op_2 = put_down_A,
Op_3 = stack_A_B,

and are then used to form two macros of length two: M_1 = (Op_1,Op_2) and M_2 = (Op_1,Op_3).

For each of the three operators, the precondition and DEL/ADD lists are given, and the postcondition is computed from the latter:

Op_1 = pick_up_A
P_1 = {clear_A, on_table_A, arm_empty}
D_1 = {on_table_A, arm_empty}
A_1 = {holding_A}
$Q_1 = (P_1 - D_1) \cup A_1$
= {clear_A, holding_A}

Op_2 = put_down_A
P_2 = {holding_A}
D_2 = {holding_A}
A_2 = {on_table_A, arm_empty}
$Q_2 = (P_2 - D_2) \cup A_2$
= {on_table_A, arm_empty}

Op_3 = stack_A_B
P_3 = {holding_A, clear_B}
D_3 = {holding_A, clear_B}
A_3 = {on_A_B, arm_empty}
$Q_3 = (P_3 - D_3) \cup A_3$
= {on_A_B, arm_empty}.

Using equation (6), we have:

$P_{1,2} = P_1 \cup (P_2 - A_1)$
= {clear_A, on_table_A, arm_empty}

and using (7) and (3), respectively:

$D_{1,2} = (D_1 \cup D_2) - (A_2 \cup \neg P_{1,2})$
= {on_table_A, arm_empty, holding_A}
– {on_table_A, arm_empty}
= {holding_A}

$A_{1,2} = [(A_1 - D_2) \cup A_2] - P_{1,2}$
= [{ } ∪ {on_table_A, arm_empty}]
– {clear_A, on_table_A, arm_empty}
= { }.

Here, we expected to obtain empty lists because the macro leaves the model in its initial state. The question is then: why the literal "holding_A" is present in $D_{1,2}$?

"We" know that in the blocks world, if "arm_empty" is true, then "holding_A" is false. So, "we" can conclude that "holding_A" is always false before the execution of Op_1 or of (Op_1,Op_2), so it does not have to be deleted (i.e., to be present in $D_{1,2}$).

"We" know this by the implicit use of the constraint
¬holding_A <= arm_empty.

The equations developed previously do not take into account such constraints and axioms, and thus are not expected to produce correct results when they are present in the theory. However, we can rewrite the semantics of previous tasks by compiling this axiomatic knowledge into preconditions, DEL/ADD lists and postconditions, thus removing the implicit reference, and by doing so, ensuring the coherence of the state[1]. In the worst case, lists are two times longer, due to the explicit handling of the closed-world assumption:

Op_1 = pick_up_A
P_1 = {clear_A, on_table_A, arm_empty, ¬holding_A}
D_1 = {on_table_A, arm_empty, ¬holding_A}
A_1 = {¬on_table_A, ¬arm_empty, holding_A}
$Q_1 = (P_1 - D_1) \cup A_1$
= {clear_A,¬on_table_A,¬arm_empty,holding_A}

Op_2 = put_down_A
P_2 = {holding_A, ¬arm_empty, ¬on_table_A}
D_2 = {holding_A, ¬arm_empty, ¬on_table_A}
A_2 = {¬holding_A,arm_empty,on_table_A}
$Q_2 = (P_2 - D_2) \cup A_2$
= {¬holding_A, arm_empty, on_table_A}

Op_3 = stack_A_B
P_3 = {holding_A, clear_B, ¬arm_empty}
D_3 = {holding_A, clear_B, ¬arm_empty}
A_3 = {¬holding_A, ¬clear_B, arm_empty, on_A_B}
$Q_3 = (P_3 - D_3) \cup A_3$
= {¬holding_A, ¬clear_B, arm_empty, on_A_B}

With this new representation, we obtain for $M_1 = (Op_1,Op_2)$:

$P_{1,2} = P_1 \cup (P_2 - A_1)$
= {clear_A, on_table_A, arm_empty, ¬holding_A} ∪ { }
= {clear_A, on_table_A, arm_empty, ¬holding_A}

$D_{1,2} = (D_1 \cup D_2) - (A_2 \cup \neg P_{1,2})$
= {on_table_A, arm_empty, ¬holding_A, holding_A, ¬arm_empty, ¬on_table_A}
– {¬holding_A, arm_empty, on_table_A, ¬clear_A, ¬on_table_A, ¬arm_empty, holding_A}
= { }

$A_{1,2} = [(A_1 - D_2) \cup A_2] - P_{1,2}$
= [({¬on_table_A, ¬arm_empty, holding_A} – {holding_A, ¬arm_empty, ¬on_table_A}) ∪ A_2] – $P_{1,2}$
= [{ } ∪ {¬holding_A, arm_empty, on_table_A}] – $P_{1,2}$
= {¬holding_A, arm_empty, on_table_A} – {clear_A, on_table_A, arm_empty, ¬holding_A}
= { }

$Q_{1,2} = (P_{1,2} - D_{1,2}] \cup A_{1,2} = P_{1,2}$.

Similarly, omitting detailed calculations, we obtain for $M_2 = (Op_1,Op_3)$:

$P_{1,3}$ = {clear_A, on_table_A, arm_empty, ¬holding_A, clear_B}
$D_{1,3}$ = {on_table_A, clear_B}
$A_{1,3}$ = {¬on_table_A, ¬clear_B, on_A_B}
$Q_{1,3}$ = {clear_A, arm_empty, ¬holding_A, ¬on_table_A,¬clear_B, on_A_B}.

[1] There are two ways to obtain properties of the states which take into account knowledge represented by the axioms. One way is to infer properties using axioms in the inference process. The other way is to compile the axiomatic knowledge into DEL and ADD lists as described above (and presented in more details in Section 4).

4 Extending the TWEAK$^+$ Formalism

The main advantage of TWEAK$^+$, as well as in STRIPS, is the STRIPS assumption avoiding the specification of frame axioms, i.e., the requirement that the DEL and ADD lists of an operator specify everything about the initiating state that is altered by the execution of the operator. For domains more complex than the one in Section 3, such an enumeration can be tedious, or even impossible. For instance, if B is a consequence of A, then every operator having A in its ADD list must also have B in it.

One way to overcome the problem ([Fahlman 1974; Fikes 1975]) consists of separating the literals into two (not necessarily disjoint) classes, the primitive (or primary) and the inferential (or secondary) literals. Only the effects on the primitive literals are specified inside the DEL/ADD lists, and an axiomatic theory describing how the inferential literals are defined in terms of primitive ones is provided. As [Waldinger 1977] pointed out, doing so makes easier the description of operators, allows more efficient updates of the states, and makes possible the introduction of new relationships between literals without modifying the operators' descriptions.

This is the kind of extensions we will consider here. As we will see, having axiomatic theory makes invalid the bit vectors equations in certain situations. We will characterize these situations and propose methods (although less efficient that the bit vector equations) to compute a description for a macro.

Even if an axiomatic theory is used, it may be impossible to satisfy the STRIPS assumption, because an operator may have effects that cannot be fully determined from its precondition and its DEL/ADD lists. We call such an operator non-deterministic; in the present context, we consider two related sources of non-determinism:

(1) the operator has effects not fully described by its definition (called side effects);
(2) the operator can be executed in several ways (i.e., it can be expanded into more than one sequence (called expansion or linearization) of primitive operators).

Typically, the first source of non-determinism comes from the impossibility of providing all the effects via DEL/ADD lists, due to the non-determinism of the domain.

The second source of non-determinism arises when a macro is allowed to contain not only operators (and possibly other macros) but also goals, and when more than one macro can be executed to achieve one of these goals. In such a case, the macro has more than one expansion and the description (i.e., the precondition and the DEL/ADD lists) may differ for each expansion, thus making the macro non-deterministic.

To take into account these two types of non-determinism in the description of operators, and to guarantee that the description is "conservative" (in the sense that all that is specified happens), the following definitions are proposed:

wk-precond(Op) =
the weakest condition guaranteeing the complete execution of Op

st-postcond(Op) =
the strongest condition obtained for all expansions of Op when executed under precond(Op).

In this context, "to be conservative" means to prefer having an eventually too strong precondition or too weak postcondition rather than having an incorrect operator description. One can look at data-flow analysis as an analogy, where useful properties of a sequence of statements are computed [Aho *et al.* 1986], and where to be "conservative" means to prefer missing optimization opportunities to guarantee no changes in what the program computes.

Now, consider that TWEAK$^+$ is augmented with an axiomatic theory where axioms are implication formulas in the disjunctive normal form "$c <= p_1 \vee p_2 \vee ... \vee p_n$", where p_i's are products (conjunctions) of literals, c is a consequent literal, and where all literals are separated into two disjoint classes: primitive literals (those appearing only in the antecedent part of axioms) and inferential literals (those appearing in the consequent part of at least one axiom).

Using a reduction procedure (as the one described in [Ginsberg 1988]), such an axiomatic theory can be reduced to a logically equivalent theory in disjunctive normal form where only primitive literals appear in the antecedent part. Then, each inferential literal appearing in the description of an operator can be replaced with its "reduction" containing only primitive literals. At this point, the axiomatic theory can be eliminated, and what remains is a set of operator descriptions logically equivalent to the initial set of descriptions, and containing only primitive literals.

There are two disjoint possibilities: (1) the new operator descriptions contain only conjunctions, (2) at least one operator description contains a disjunction.

In the first case, the new operator descriptions conform with the TWEAK$^+$ formalism, and the bit vector equations can be applied. In the second case, at least one of the new operators cannot be represented in TWEAK$^+$ because the latter formalism does not support disjunctions, and the equations cannot be used.

The disjunction may originate from two sources: (1) the disjunction was present in the axiomatic theory and was transferred into the operator description, (2) the disjunction comes from the negation, in the operator description, of a inferential literal whose reduction contains a conjunction. As these two sources are independent, restrictions on both the theory and the operator descriptions must be made to ensure that no disjunctions will be present in the reduced operator descriptions. These restrictions are the following:

(1) disjunctions are not allowed in the axiomatic theory
(2) negations of inferential literals are not allowed inside operator descriptions.

With these restrictions, the descriptions of macros can be determined using the bit vector equations presented in Section 2. Unfortunately, these constraints are too limiting for most of the interesting domains, and alternative approaches must be taken to derive the description of macros. For instance, the following approach can be taken

to handle disjunctions (or any arbitrary formulas for the preconditions):

1. identify an initial condition C where a macro Op is executable;
2. use EBG to identify a precondition C' = wk-precond(Op) of the macro, determined by the condition C specified in 1;
3. obtain the postcondition C" = st-postcond(Op) of the macro for C by executing the macro on C'
4. do 1, 2, 3 for several examples with the same macro on different initial conditions C; obtain several general macro descriptions that differ only by their conditions C' and C" (not their operators).

As an illustration, given the operator descriptions

$$\text{precond}(Op_1) = \{p\} \qquad \text{postcond}(Op_1) = \{p,q\}$$
$$\text{precond}(Op_2) = \{q,r\} \qquad \text{postcond}(Op_2) = \{q,r,s\}$$

and the macro $Op = (Op_1, Op_2)$, the step 1 identifies that C must contain p in order to make Op_1 executable, and must also contain r to make Op_2 executable (the condition q needed by $\text{precond}(Op_2)$ is already provided by $\text{postcond}(Op_1)$ and does not have to be present in precond(Op)). The result of step 1 is thus C = {p,r}. In the present example, step 2 produces C' = C = {p,r} (no generalization), and step 3, by executing Op under {p,r}, produces the condition {p,q,r,s}. Because step 4 (repeating steps 1-2-3) produces no new conditions in the present case, the description computed for the macro Op is:

wk-precond(Op) = {p,r} st-postcond(Op) = {p,q,r,s}.

The above approach computes a "sufficient" precondition (not always "necessary") for a macro. Note that the description of more complex macros using the macro obtained with this method cannot be computed using bit vector equations; however, the same approach can be used to compute a sufficient condition of such complex macros.

Other approaches can be used to compute a description of a macro in non-deterministic domains using an axiomatic theory. For instance, [Pelletier 1992] describes a non-deterministic method for determining the precondition wk-precond(Op) of a macro, and from it, its postcondition st-postcond(Op). The precondition of a macro is found by cumulating the parts of the precondition of the operators that fail to be satisfied. The postcondition is found by applying in sequence the operators constituting the macro.

5 Conclusion

We have presented TWEAK$^+$, a STRIPS-like formalism for the description of operators and macros constructed by chaining operators. The essential extension of the STRIPS formalism consists of allowing negative literals in the DEL and ADD lists. Consequently, in TWEAK$^+$, each operator or macro is described by four lists of positive or negative literals: a precondition, a DEL list, an ADD list and a postcondition. In TWEAK$^+$ it becomes possible to specify that a given literal becomes false under the application of some operator. Such an extension, conveniently allowing us to bypass the "frame problem" that restricts and complicates the use of the traditional STRIPS representation, presents certain problems in building macros. We have shown how the description of macros can be efficiently computed, in some situations, using bit vector equations (in linear, or even sub-linear time [Aho *et al.* 1974], and linear space, with respect to the number of operators contained in the macro). These situations are characterized by deterministic domains; if the domain also uses an axiomatic theory, the equivalent reduced operator descriptions must not contain disjunctions.

For the other situations, the bit vector equations cannot be used, and heuristical methods to compute sufficient precondition and postcondition for macros were proposed.

In future work, we will try to clarify part of our motivation, i.e. under what circumstances is one interested in building macros, as opposed to applying derivational analogy or using search control knowledge. Suppose that a domain, a planner and a library of existing plans are given. We will attempt to obtain criteria which can be applied to the domain in order to select, for the planner, the "best" mix of methods of re-using plans from the library. The notion of "best" includes several possible criteria, e.g., the amount of search necessary to build the plan, or the memory retrieval effort necessary to locate plans for potential reuse in the library. The problem is described in more detail in [Langley *et al.* 1992].

Acknowledgements

The work described here has been supported by the Natural Sciences and Engineering Research Council of Canada, the Government of Ontario (URIF and OTF Programs), the Department of Systems and Computer Engineering of Carleton University, Cognos Inc, and the Canada Centre for Remote Sensing.

References

[Aho *et al.* 1974] Alfred V. Aho, J. E. Hopcroft and Jeffrey D. Ullman. The Design and Analysis of Computer Algorithms. Addison-Wesley Publishing Company, 1974.

[Aho *et al.* 1986] Alfred V. Aho, Ravi Sethi and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley Publishing Company, 1986.

[Chapman 1987] David Chapman. Planning for Conjunctive Goals. *Artificial Intelligence*, 32:333-377, 1987.

[Fahlman 1974] S. E. Fahlman. A Planning System for Robot Construction Tasks. *Artificial Intelligence*, (5):1-49, 1974.

[Fikes 1975] Richard E. Fikes. Deductive Retrieval Mechanisms for State Description Models. In Proceedings *of the Fourth Internationale Joint Conference on Artificial Intelligence*, pages 99-106, Tbilisi, Georgia, USSR, 1975.

[Fikes *et al.* 1972] Richard E. Fikes, Peter E. Hart and Nils J. Nilsson. Learning and Executing Generalized Robot Plans. *Artificial Intelligence*, 3(4):251-288, 1972.

[Fikes and Nilsson 1971] Richard E. Fikes and Nils J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, (2):198-208, 1971.

[Genesereth and Nilsson 1987] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, Inc, Los Alto, California, 1987.

[Georgeff 1987] Michael P. Georgeff. Planning. *Annual Review of Computer Science*, 2:359-400, 1987.

[Ginsberg 1988] Allen Ginsberg. Knowledge Base Reduction: A New Approach to Checking Knowledge Bases for Inconsistency and Redundancy. In Proceedings *of the Seventh National Conference on Artificial Intelligence*, pages 585-589, Mineapolis, 1988.

[Kambhampati and Hendler 1990] Subbarao Kambhampati and James A. Hendler. A Validation Structure Based Theory of Plan Modification and Reuse. *Artificial Intelligence*, (to appear).

[Korf 1985] R. Korf. Macro-operators: A Weak Method for Learning. *Artificial Intelligence*, (26):35-77, 1985.

[Langley *et al.* 1992] Pat Langley, Stan Matwin and J. A. Allen. Knowledge and Regularity in Planning. In Proceedings *of the 1992 AAAI Spring Symposium on Computational Considerations in Supporting Incremental Modification*, (to appear).

[Mitchell *et al.* 1986] Tom M. Mitchell, R. M. Keller and S. T. Kedar-Cabelli. Explanation-Based Generalization: A Unifying View. *Machine Learning*, 1(1):47-80, 1986.

[Pelletier 1992] Bertrand Pelletier. Unsupervised Learning From a Goal-Driven Agent. Ph. D. Thesis (in preparation), Department of Systems and Computer Engineering, Carleton University, 1992.

[Prieditis 1990] Armand E. Prieditis. Discovering Algorithms from Weak Methods. In *Machine Learning*, vol. III, pages 351-359, 1990.

[Rich 1983] Elaine Rich. *Artificial Intelligence*. Series in Artificial Intelligence. McGraw-Hill, 1983.

[Tadepalli 1991] Prasad Tadepalli. Learning with Inscrutable Theories. In Proceedings *of the Eighth International Workshop on Machine Learning*, pages 544-548, Evanston, California, 1991. Morgan Kaufmann.

[Waldinger 1977] Richard J. Waldinger. Achieving Several Goals Simultaneously. In *Machine Intelligence 8: Machine Representations of Knowledge*, pages 94-136. Ellis Horwood, Chichester, UK, 1977.

ARTIFICIAL INTELLIGENCE IN THE REAL WORLD: A CRITICAL PERSPECTIVE

Richard S. Rosenberg
Department of Computer Science
University of British Columbia
Vancouver, B. C. V6T 1Z2
rosen@cs.ubc.ca

Abstract

This paper presents an introduction to a number of social issues which may arise as a result of the diffusion of Artificial Intelligence (AI) applications from the laboratory into the workplace and marketplace. Such applications include expert systems (ES), image processing, robotics, and natural language understanding. Of the many social issues of concern, four are selected for treatment here as representative of other potential problems likely to follow such a powerful technology as AI. These four are work (how much and of what kind), privacy (on which the assault continues), decision-making (by whom and for whose benefit), and social organization (how in a society in which intelligent systems perform so many functions). Finally it is argued that both a major programme of study in this field be launched and that practitioners assume the responsibility to inform the public about their work.

Keywords

artificial intelligence, applications, social issues, work, privacy, responsibility

1. Introduction

A few years ago, the question, "What is Artificial Intelligence?" was frequently heard. Clearly that situation does not obtain today. From a somewhat esoteric subdiscipline of Computer Science, AI has emerged as a major force in the universities, the marketplace, and the United States defense establishment. Many professors have formed companies to market products emerging from their research. Such corporations as Texas Instruments, Xerox, General Motors, Hewlett-Packard, Schlumberger, and IBM have launched major research and development programs. Industry and the popular media are rife with such terms as expert systems (ES), natural language interfaces (NLI), knowledge engineering, image understanding, and intelligent robots. AI has well and truly arrived.[1]

Note that even from its earliest days, some thirty-five years ago, two paths have characterized research in AI. One represented the research aim of trying to understand the nature of intelligent behaviour - simply put, the scientific approach. The other is motivated by the attempt to design and build working systems which perform useful tasks normally said to require intelligence - the engineering approach. However, the importance of this distinction should not be overstated. Usually the scientific and engineering approaches co-exist, with some tension, as researchers of both persuasions confront similar problems and influence directions in the field. Nevertheless, in the haste to deliver marketable products, the danger is that science will suffer. For one thing, the effort necessary to solve a wide range of fundamental problems will be diverted to the more immediate needs of commercial enterprises. Of course, the reputation of the field as whole will suffer if exaggerated claims are not realized.

AI is a vital and exciting field and it has attracted many scientists. Research at universities, private laboratories, and government installations is proceeding across many fronts including knowledge representation, reasoning, problem solving, natural language understanding, image processing, expert systems, logic programming and heuristic search, among others. A public discussion of these and other areas was stimulated by an international conference held in Tokyo, in October of 1981, to which about 90 distinguished foreign scientists were invited. At that time Japan announced the initiation of its Fifth Generation Project, an ambitious program to develop useful and powerful intelligent computers. Other nations were invited to co-operate, but mindful of Japan's worldwide lead in consumer electronics, they decided to launch their own national and regional projects. The response of the United States, the present world leader in computers in general and AI in particular, is obviously of prime importance. One significant action by the U.S. Department of Defense was the Strategic Computing Initiative (not Star Wars) launched in 1983, which has

had, and continues to have, serious implications for AI research and researchers, given the massive amounts of funds to be spent.

All the foregoing represents a prologue (no pun intended) for a preliminary assessment of the implications of successful AI research on society. It should be noted, however, that even partially realized AI systems will have considerable impact. Among the dimensions of this investigation are such issues as how achievements in AI will affect human self-worth especially if one result of the diffusion of intelligent computers is a net reduction in employment. Such a reduction could come about through advanced industrial automation including robots and, in the office, through the realization of a true Office of the Future. How will the workplace itself be altered as humans cope with 'intelligent', non-human co-workers? What about the role of such systems in medical treatment, education, financial services, and many other areas of human experience. Other serious issues relate to their use in military decision-making, law enforcement and surveillance, and to the more general concerns of privacy itself.

Any treatment of the impact of AI on society must be situated in the more general concerns of the impact of technology itself. However the nature of AI suggests that as a new technology, its influence may not be merely quantitative, not just more of the same, not just the latest improvement in the continuous progress of the industrial revolution. If its potential is realized, AI may usher in a new age, and it will truly be a qualitative break with the past. In what follows, a number of issues will be presented which form the basis of a proposed research programme. Some possible social concerns are elaborated in the following section under the categories work, privacy, decision-making, and social organization. The paper concludes with a call to AI researchers to speak openly and forcefully about their work and its potential for good or ill.

2. An Overview of Some Issues

As discussed above, the term AI has now entered the public consciousness. Articles describing computers which "think" appear regularly in major magazines. In popular computer magazines, AI is a regular feature as languages (Lisp, Prolog), expert system shells (M1,KEE), speech recognition, vision, and natural language understanding are evaluated. Does this imply that AI, as manifested in working computer programs has been fully realized? Of course not. Certainly the mystique of AI, the idea of steady progress towards computers able to perform a wide range of human-like behaviour has been established in the public psyche.

The publicity and the hype are definitely upbeat and expansive and herald a golden age of leisure. All work is to be done by machines, enabling people to realize their potential, free from worry about jobs, rent, and food. Many questions naturally arise, aside from the obvious one about whether or not intelligent computers are really possible.[2] Winner [1986] calls for the development of a philosophy of technology, "to examine critically the nature and significance of artificial aids to human activity" (p. 4). If those artificial aids purport to be intelligent, how much more the urgency of the enterprise. Of course, Winner himself is quite skeptical about both the achievements of, and the claims for, AI. He notes that, ".. children have always fantasized that their dolls were alive and talking" (p. 14).

Clearly there are many applications of intelligent systems of direct and unequivocal benefit to society at large and workers in particular. For example dangerous activities in mines, under the sea, in nuclear plants, in the chemical industry, and elsewhere are prime candidates for robots. Less dangerous, but obviously unpleasant, jobs should in the near future also be phased out as exclusively human preserves. There are many ways that this process can take place but if history has anything to teach in this area it is that the protection of workers is not usually the major reason that change takes place. Lessons drawn from the introduction of computer technology into the workplace by Noble [1984], Shaiken [1984], Kraft [1977], Garson [1988], and others point out the existence of alternative strategies and the motives of managers in constraining them.

Of the many issues to consider in this context let me suggest the following ones, not all of which will be discussed here:

1. A realistic evaluation must be attempted of current and near-future prospects for AI applications at home, in the workplace, and in the government.
2. A similar evaluation is necessary of the impact of computer-related technology in the workplace, balancing benefits against perceived problems, including deskilling, monitoring, job loss, restricted promotion paths, breakdown of traditional social organizations in the office, limited entry level opportunities, and health-related concerns.
3. The implications of partially realized intelligent systems in terms of the requirements placed on humans to accommodate to their, the systems', inadequacies must be considered. In the haste to introduce AI into the workplace, pressures may be placed on people to work with systems, which, while advertised as intelligent, are seriously deficient in many areas.
4. Of particular interest is the role of AI in decision-making, whether in financial institutions, in the executive suite, or in diverse military situations such as autonomous land vehicles, pilots' aid, aircraft carrier battle management (all of which are components of the Strategic Computer Initiative launched in 1983) or in the evaluation of possible nuclear attack (either in or out of

the context of SDI, the Strategic Defense Initiative).

5. Intelligent systems may find ready application in intelligence activities such as automatic interpretation of tape recordings and the cross-correlation of electronic files. Added to current threats to privacy, the availability of such powerful mechanisms could increase real and anticipated assaults on individual privacy.

6. Futuristic projections of a society without poverty brought about by the extensive diffusion of AI applications have been considered by science fiction writers and futurologists and more recently by AI researchers themselves, including Albus[1976; 1983) and Nilsson [1984). Even Wassily Leontief [Leontieff and Duchin, 1986], a Noble winner in Economics, has been concerned about such a future.

Speculation is interesting but the assumptions underlying the forecasts must be carefully analyzed. Questions to be considered include the following:

> What replaces regular work as a necessary part of life?
>
> How is wealth to be distributed if a wage system is no longer operative?
>
> How will the political structure respond?
>
> How will human dignity and self-worth be affected if we are no longer defined in great part by what we do?

It should be kept in mind that these questions are obviously so difficult to answer, or even to characterize, that only a beginning is made here. However, it is important that they be raised and that a serious discussion be initiated.

3. Artificial Intelligence and Society

Obviously space does not permit a detailed exposition of the many issues associated with the explosive growth of computers but given the revolutionary possibilities of AI, it is worthwhile to suggest those areas, in society, most likely to be affected. The following will focus on work, privacy, decision-making, and social organization.

Work

Of particular concern is the impact of computers on work - both the nature of the job itself and the number of jobs. The relation between technology and work is complicated and operates on many dimensions. The economic imperative to increase productivity by the introduction of new technology is alive and well today, as it has been since the onset of the Industrial Revolution and even before. However, the emergence of AI will bring to the fore the question: Will there be a massive loss of jobs and if so, what kinds of jobs will be available? Various writers, including more recently AI researchers, such as Nilsson [1984], have speculated about a future in which intelligent machines produce the goods and provide many of our services. Of course this is also a theme explored by many science fiction writers from both a utopian and dystopian point of view. It is clear that serious issues of income distribution, self worth, and the basic political organization of society are involved. If the means of production gradually move from human hands to robotic ones, then the distribution of wealth may indeed become an urgent social concern rather than an accepted, intrinsic and automatic process.

If the foregoing appears to be too speculative and far-fetched then perhaps more immediate concerns of the changing nature of work itself might be considered. The introduction of robots (or more general forms of industrial automation) into the factory or computer networks and associated equipment into the office, has already had, and will most certainly have, a wide impact on workers. Among the problems already identified are deskilling, monitoring, health effects, psychological stress, and the issues of self-worth and dignity. Others are restriction of promotion paths, the breakdown of existing workplace social organization, and a limitation on entry-level jobs. These are discussed in Rosenberg [1986; 1992] as well as in many other sources. The incorporation of AI technology into robots and office automation can only exacerbate these problems, in most cases, but there are clear benefits in others. Weitz [1990, p. 50] argues that at worst ES will have little impact on work and it is more likely that they will contribute to "the evolution of the lean, flexible, knowledge-intensive, postindustrial organization."

Surely intelligent machines can perform tasks undesirable for people, for example, both underground and undersea mining, dealing with hazardous wastes, welding and spray painting, and handling noxious gases. Of greater significance is the intellectual benefit of intelligent systems for improving efficiency in every aspect of human endeavour. Systems which never forget, are repositories of up-to-date knowledge, and which are responsive in a variety of ways to human abilities and needs, should improve quality, reduce errors, alleviate stress, and contribute to an efficient and productive economy, in which it may be hoped that the benefits will be more equitably distributed. Intelligent aids to information retrieval, decision-making, planning, and problem-solving are appearing and will continue to improve. For those holding jobs which require, at their core, the ability to make decisions, AI augmented systems offer both hope and despair. Hope exists in the potential power of the new systems to "amplify intelligence," to offer to the mind what motors have offered to muscles. Despair lurks in the threat to human autonomy, to the very essence of what makes us human - our ability to reason about the world and control our own destiny.

Even unrealized AI may project such a threat.

Schefe [1990] warns that the negative work-related effects of automation may be exaggerated by ES especially if little attention is paid to them by the leading developers. It is important that those who will be working with ES be educated and made aware of the abilities and limitations of the technology and not be treated as passive agents soon to go the way of their one-time fellow employees.

Privacy

The growing threat to individual privacy represented by the increasing use of computer databases has long been recognized by civil libertarians as well as the general public. The existence of private databases containing employment, credit, cable, and medical records, among others, and public databases storing tax, census, education, and voters' records, creates the possibility for abuse because of the ease of accessing these records under a variety of search conditions. Such terms as computer profiling and computer matching have become quite common recently, as well as controversial. The former relates to the attempt to predict behaviour, of a potentially criminal kind, by defining a profile which can then be searched for in existing files, thus identifying individuals "likely" to exhibit such behaviour. Computer matching has already been employed in many situations to cross index files in order to determine whether or not individuals have committed crimes as revealed by inconsistencies in their records. For example, an examination of property records might reveal that Mr. X has received considerable income on the sale of a piece of land while at the same time collecting unemployment insurance, as determined from an examination of payments listed in the files.

Both of these practices have aroused considerable debate as they tend to presume a priori guilt for some individuals, who are to be identified through a search for confirming evidence in the computer matching case and subsequently placed under surveillance in the profile case. Such actions seem to be a violation of presumed innocence and may place individuals under jeopardy. The use of AI can only encourage wider applications of such procedures and even the development of new techniques of investigation. ES and AI are being used by police forces in investigations, including surveillance, and is also being used in the business world for similar purposes.

Currently, computer monitoring or surveillance, in the workplace, has become a growing concern with respect to the rights of workers. It is a trivial task for operating systems to record keystroke counts and thereby determine if a worker's output has deviated from pre-established norms. Such computer monitoring is only one weapon in management's arsenal for measuring, testing, and monitoring employees. Others include the use of TV cameras, telephone eavesdropping, drug testing, genetic screening, and polygraph testing [*Supervisor*, 1987]. AI is beginning to play a role in more sophisticated monitoring systems that can be used to detect unusual computing behaviour. Ostensibly such "intrusion detection systems" are intended to employ AI techniques to compare individual activities with pre-stored histories, in order to identify possible illegal users or actions. Marc Rotenberg, director of the Washington office of Computer Professionals for Social Responsibility, has pointed out two crucial issues: "whether the monitoring required by such systems violates employees' rights and whether such security efforts have a deleterious effect on work [Kerr, 1990]. Note that the role of AI is to create user profiles from records of past behaviour and then to monitor deviations from these profiles.

But it must be remembered that personal privacy has been under assault ever since records have been kept. This assault has intensified since the computer and associated databases have become readily available. We live in age in which information about individuals has become a valuable commodity. The capture and use of this information by direct marketing companies has emerged as a serious threat to individual privacy. Governments in many countries and at many levels have enacted laws to deal with the most egregious abuses but various threats remain and the potential problems posed by advances in AI have created new challenges to those concerned with civil liberties in general and privacy in particular.

Decision-Making

This term will be used here in an all-encompassing sense to cover activities regularly carried out by individuals, companies, institutions, and governments, involving the assimilation of information, its organization, and finally its employment based on experience, special knowledge, theory, and perhaps, even intuition. It hardly needs remarking that every aspect of life involves decisions, whether made by the individual or made for him, or her, by others. As such, decision-making represents a fundamental component of human existence and threats to human autonomy, however couched in friendly terms, are of serious concern. In any discussion of ES, the question of the dogmatic aspect of formalized expertise can be raised. By their very nature, ES purport to capture, formalize, and disseminate expertise. From a negative point of view, the effects of this process may include, standardization, homogenization, centralization, legitimization, and a definite sense of authority and control. Part of the concern is that the formalization of knowledge as an ES for some restricted domain can be taken as the representation of the knowledge. An analogy might be the legal code, where crime is defined by

the existing relevant statutes.

Of the many applications of ES, those in financial planning including stock market interventions are increasingly critical. Many will recall Black Monday and the crash of 1987, October 19 and 20 when the greatest plunge in market prices in history occurred. The blame was immediately assigned to computers in their role as automatic initiators of stock transactions. (See Rosenberg [1992, 278-280].) Subsequent analysis demonstrated that the situation was far more complicated and that computers were only one factor, however important. Nevertheless, computers, or rather the strategies implemented on them, did bear a major responsibility. (See Lucas and Schwartz [1989].) The implications of such a memorable event are obvious, at least to government and market regulators, at least - computer trading may introduce unpredictable instabilities into the volatile operations of the stock market. What of the future? As stock market ES compete against one another, who will be the winners and who will be the losers?

In many areas of life there is no consensus, no received view and in the opinion of many, there will never be. Thus the threat posed by the rapid diffusion of ES is the limitation of diversity and the imposition of the equivalent of a state religion, with its fixed, dogmatic world view. Further, the objectification of knowledge and experience brings apparent certainty to uncertainty, rules to govern feelings, and regularity where it is unwarranted and perhaps impossible. Some of these fears may seem extreme but underlying them are serious philosophical and social concerns. For example, the by now traditional use of computers to reinforce bureaucratic decisions by appeal to an infallible computer can only be extended by the adoption of even more powerful systems, namely ES. Of course in the opinion of some critics [Dreyfus and Dreyfus, 1986] ES are fundamentally limited in the degree to which they can actually capture expertise. As such their use will impose false and possibly harmful constraints in many aspects of human experience.

The science fiction version of computer decision making has become a popular theme, with the negative aspects explored by such authors as Jack Williamson [1963; 1981] and Jack Chalker [1986]. Briefly, the premise is that massive computer systems have been given, or even take powers, in order to prevent humans from destroying themselves. Unfortunately, for humans, the system interprets its mission so literally with the result that people find their lives without challenge or purpose. It becomes impossible to exercise even a modicum of meaningful independent behaviour. That such a future fascinates science writers does not make it in any way inevitable, of course, but it does raise some interesting questions. Indeed, the increasing use of computers may result in a decrease in human decision making and the role of AI will be to accelerate this trend, especially in more critical situations.

In this respect, the debate over the Strategic Defense Initiative (SDI, or more commonly, Star Wars) has frequently turned on the question of whether or not the very large software component could perform as required. This system, expected to be an order of magnitude larger than Unix, will monitor information gathering devices, assimilate the information, decide on a response, coordinate the response, and continue with these activities until the end. AI is certainly expected to play a role in this system as one of its harshest critics, Parnas [1985], has noted in a dissenting commentary. Note that the issue of computer decision making does not begin with AI but rather that the reliance on AI may exacerbate the potential problem in a fundamental way.

This reliance on AI is being pursued in other areas as well including the development of the autonomous land vehicle and the battleship management system. The former can be thought of as an armoured, autonomous vehicle able to navigate the battlefield, avoid obstacles, and report on conditions. Its successful development requires advanced image understanding, problem solving, and decision making. The battleship system is seen as an intelligent aid to a naval commander, under engagement, who must deal with many simultaneous events. Here is a system, upon which a commander will rely in dangerous situations, perhaps putting his men in some jeopardy. But then would they be better off without the benefits possibly available from sophisticated software? A chilling, partial answer to this question occurred in July 1988 when an Iranian passenger airplane was shot down in the Gulf of Arabia by the U.S.S. Vincennes, a cruiser boasting the most sophisticated electronics equipment available (including computers of course), yet unable to distinguish this plane from a much smaller fighter aircraft.

For further discussion of the role of AI in military applications see Andriole and Hopple [1988] and Din [1987]. The latter also provides a number of papers on arms control analysis, not a very urgent issue in 1992 - fortunately.

Social Organization

How will society, or better its political institutions, respond to a future in which basic needs, both goods and services, are met by machines? Robots and advanced industrial automation are gradually reducing the blue collar workforce and this is taking place largely without AI. Changes are occurring much more slowly within the office but as we have noted above, the impact of successful developments in AI, especially in speech understanding, can result in many fewer jobs. Thus the issue, for the future may be what will replace work in most people's lives both as a means to acquire money and as a major component in the definition of self worth. However

desirable the anticipated high-tech future is, and this may itself be debatable, the means for achieving it are rarely spelled out, as most visions of utopia neglect to describe how society will make the torturous trip from the current world to the promised land.

Since the most important way to distribute the real wealth in society is through wages or salary (ignoring stocks, bonds, real estate, etc.) and the envisioned future includes a considerably reduced workforce, two questions emerge: How do people acquire goods and services beyond immediate basic needs and what replaces work, with all its trappings, in most people's lives? Nilsson [1984], Albus [1976; 1983], and Leontieff [1986] suggest partial answers to these questions. For Leontieff one response is that the average work week will have to be reduced so that people will work less but not at lower salaries, thus maintaining their earning power by sharing in the increased wealth produced by the advanced technology. For Albus, the new technology, represented by robots, will, by some undefined process, be owned, in part, by the very workers that they displace thereby providing for these workers a share of the wealth that the robots earn. Albus has referred to his vision as People's Capitalism [Albus, 1976], which others may refer to as socialism, or even communism.

Nilsson [1984], a major figure in the AI community, paints an enticing vision of a future free from drudgery, made possible by AI. Removing the need to work as the primary means to satisfy wants will permit people to realize their potential by doing what they really want to do. Thus the apparent psychological need to work will prove illusory as technology liberates society from such a mundane requirement. Nilsson offers answers to the question of how wealth is to be distributed by quoting several authors, including Albus but does not see it as a serious problem. Except for some temporary problems during the transition period, the benefits will far outweigh the difficulties. The vision is almost purely utopian.

I would argue that crucial problems have been swept aside in the enthusiasm of describing the wonders to be brought about by AI. It is not clear that any of the proposed schemes for distributing the wealth resulting from AI will indeed work given that the current political system in operation in the most technically advanced countries is capitalism, unfettered free enterprise. How societies are supposed to move from such an economic system to a system in which accumulated wealth is distributed to individuals independent of their direct responsibility for earning this wealth remains a mystery. Up to now the earnings from natural resources such as minerals, oil, coal, lumber, fish, and the land itself, accrue only to those who extract it, fish for it, or farm it, with a rather small portion returning to the state in the form of license fees and royalties. In fact if the state received a greater portion, entrepreneurs would be discouraged from searching for new resources. Based on this experience, what reason is there to believe that new forms of wealth achieved by the use of new technologies, such as AI, will be made universally available?

It is not at all obvious how the technologically-induced utopia will come about or even that it will. Massive changes in political systems as well as social organizations will be necessary. Work and money are just part of the equation. Autonomy, self respect, and civil liberties are others. None of these are gifts bestowed by a benevolent state, especially one which is the product of major technological innovations. It is for this reason that the process involved in moving towards a new society is so crucial and that an awareness and realistic understanding of how technology operates, perhaps the philosophy of technology, referred to earlier, is so important.

In a letter to *IEEE Expert* with the provocative title, "Do Expert Systems Threaten Democracy?" Yuval Lirov [1991] defends the goals of, and justification for, the development of such self-sufficient global information systems as Lenat's Cyc [Lenat and Guha, 1990]. Although recognizing that in the early stages such expert systems may be "antidemocratic," he goes on to say, "Fear of this phase is pathetic and its denial is dangerous. Our moral obligation is to develop tools to control our intelligent golem during this exciting phase." [Lirov, p.9] This position is a fairly typical expression of technological optimism and the belief that ultimately we (citizens, professionals, society?) are in control of our technology. Many counterexamples abound, including nuclear energy (and weapons), environmental devastation, genetic engineering, and reproductive technology abuses. The impact of technology, especially such a potentially revolutionary one as AI, on society is not predictable, nor is its conscious shaping and control particularly likely.

4. Conclusions

This paper represents one small step, among others, in an attempt to explore the social impact of such far-reaching technologies as AI and genetic engineering. If this effort were to be classified under the rubric of futurology, the major point would be lost. It is important to keep in mind that the relations among economics, politics, social organization, and technology are deep and intricate and it would be foolish to trivialize them by predicting a future based on unfettered technology. If AI is seen as a natural and expected continuation of the historical evolution of technology, then there is no reason to expect its effect to be substantially discontinuous with the past. For example, some two hundred years since the onset of the Industrial Revolution, we find that unemployment rates are still relatively low. The contribution of AI would have to be revolutionary, in the true meaning of this word, to transform the world in a way which would

result in a major decrease in real employment.

Concern with the impact of a given form of technology should not be left, in general, to "outsiders." I do not mean, of course, that practitioners should be regularly involved in the criticism business but rather that as part of their professional responsibility, both to themselves and to society, they should be prepared to speak out about their work and to inform the public about possible implications. An excellent review of many important social issues related to AI can be found in Athanasiou [1987] and Yazdani and Narayanan [1984]. Croy [1989] raises some important ethical issues with respect to the use of ES in education. I also recommend Weizenbaum's [1986] impassioned call to the computer science community, in general, and to the AI sub-community, in particular, to recognize its responsibility by considering carefully the outcome of its work. Furthermore he argues that computer professionals must be aware of how dependent society is on their expertise for future developments, especially military applications.

For those who would respond that the duty of professionals is to themselves, their discipline, their institution, and their clients and that their personal views should be kept private, I would point out that the potentially significant implications of AI demand that its researchers take an active role in initiating an ongoing public debate on the issues, in order to educate society at large. Such a demand is not unique to computer science; note that at many law schools, in the U.S., there is currently a movement afoot, called Critical Legal Studies (CLS), which seeks to instill social responsibility within the educational process. For example, in a brief statement, describing the goals of the movement, we read, "Although CLS advocates are clearer about what they oppose than propose, they favour manipulating existing law to guarantee economic equality and to eliminate distinctions based on class, race, and sex. In short they'd dispense with most legal precedents."[3]

Surely, a relatively new discipline, such as computer science, can establish for itself the practice of encouraging its practitioners to speak openly and forcefully about their work.

Notes

1. See Andrew Pollack, "Setbacks for Artificial Intelligence," *The New York Times*, March 4, 1988, First Business Page, p. 32. Pollack reports that a number of leading companies in ES development tools, Lisp machines, and ES applications lost money and instituted layoffs in 1987. Although he notes that AI has failed to live up to its promise, the major reason for these difficulties is poor financial management rather than poor products.

2. This important question will not be answered here (or anywhere else for that matter) for the issue at hand is to try to understand how the gradual diffusion of this most advanced of all technologies will affect society. On the other hand if the critics, Dreyfus [1972] and Dreyfus and Dreyfus [1986], are to be believed, computers will never be intelligent and all questions revert to the traditional study of the impact of technology in general on society.

3. See Leslie Helm and Lawrence J. Tell, "The Radical Rumblings Shaking Up Law Schools," *Business Week*, June 6, 1988, pp. 116.

References

[Andriole and Hopple, 1988] Stephen J. Andriole and Gerald W. Hopple, editors. *Defense Applications of Artificial Intelligence: Progress and Projects*. Lexington Books (D. C. Heath and Company), Lexington, MA, 1988.

[Athanasiou, 1987] Tom Athanasiou. High-Tech Politics: The Case of Artificial Intelligence. *Socialist Review*, 17(2):6-37, No. 92, March/April 1987.

[Albus, 1976] James S. Albus. *People's Capitalism*. New World Books, College Park, MD, 1976.

[Albus, 1983] James S. Albus. The Robot Revolution: An Interview with James Albus. *Communications of the ACM*, 26(3): 179-180, March 1983.

[Chalker, 1986] Jack L. Chalker. *Lords of the Middle Dark*. Ballantine Books, New York, 1986.

[Croy, 1989] Marvin J. Croy. Ethical Issues Concerning Expert Systems' Applications in Education. *AI & Society*. 3(3):209-219, July-September 1989.

[Din, 1987] Alan M. Din, editor. *Arms and Artificial Intelligence: Weapon and Arms Control Applications of Advanced Computing*.. Oxford University Press, New York, 1987.

[Dreyfus, 1972] Hubert L. Dreyfus. *What Computers Can't Do*. Harper and Row, New York, 1972.

[Dreyfus and Dreyfus, 1986] Hubert L. Dreyfus and Stuart E. Dreyfus. *Mind Over Machine: The Power of Human Intuition and Expertise in the Era of the Computer*. The Free Press, New York, 1986.

[Garson, 1988] Barbara Garson. *The Electronic Sweatshop: How Computers Are Transforming the Office of the Future into the Factory of the Past*. Simon and Schuster, New York, 1988.

[Kerr, 1990] Susan Kerr. Using AI to Improve Security. *Datamation*, 57-60, February 1, 1990.

[Kraft, 1977] Philip Kraft. *Programmers and Managers: The Routinization of Computer Programming in the United States*, Springer-Verlag, New York, 1976.

[Lenat and Guha, 1990] Douglas B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley Publishing Company, Reading, MA, 1990.

[Leontief and Duchin, 1986] Wassily Leontief and Faye Duchin. *The Future Impact of Automation on Workers*. Oxford University Press, New York, 1986.

[Lirov, 1991] Yuval Lirov. Do Expert Systems Threaten Democracy? *IEEE Expert*,, 6(2):2, 9, April 1991.

[Lucas and Schwartz, 1989] Henry C. Lucas, Jr. and Robert A. Schwartz, editors. *The Challenge of Information Technology for the Securities Markets: Liquidity, Volatility, and Global Trading*. Business One Irwin, Homewood, IL, 1989.

[Yazdani and Narayanan, 1984] Masoud Yazdani and Ajit Narayanan, editors. *Artificial Intelligence: Human Effects*. Ellis Horwood Limited, Chichester, England, 1984.

[Nilsson, 1984] Nils Nilsson. Artificial Intelligence, Employment and Income. *AI Magazine*, 5(2):5-14, Summer 1984.

[Noble, 1984] David F. Noble. *Forces of Production*. Alfred A. Knopf, New York, 1984.

[Parnas, 1985] David Lorge Parnas. Software Aspects of Strategic Defense Systems. *American Scientist*, 432-440, September-October 1985.

[Rosenberg, 1992] Richard S. Rosenberg. *The Social Impact of Computers*. Academic Press, Boston, 1992.

[Rosenberg, 1986] Richard S. Rosenberg. *Computers and the Information Society*. John Wiley & Sons, New York, 1986.

[Shaiken, 1984] Harley Shaiken. *Work Transformed: Automation and Labor in the Computer Age*. Holt, Rinehart and Winston, New York, 1984.

[Schefe, 1990] Peter Schefe. The Impact of Expert Systems on Working Life - An Assessment. *AI & Society*. 4(3):183-195, July-September 1990.

[*Supervisor*, 1987] *The Electronic Supervisor: New Technologies, New Tensions*. OTA-CIT-333, U. S. Congress, Office of Technology Assessment, Washington, D. C., September 1987.

[Weitz, 1990] Rob R. Weitz. Technology, Work, and the Organization: The Impact of Expert Systems. *AI Magazine*, 11(2):50-60, Summer 1990.

[Weizenbaum, 1986] Joseph Weizenbaum. Not Without Us. *Computers and Society*, 16(2,3): 2-7, Summer/Fall 1986

[Williamson, 1981] Jack Williamson. *The Humanoid Touch*. Bantam, New York, 1981.

[Williamson, 1983] Jack Williamson. *The Humanoids*. Lancer, New York (originally published in 1949).

[Winner, 1986] Langdon Winner. *The Whale and the Reactor*. The University of Chicago Press, Chicago, 1986.

Acknowledgment

I would like to acknowledge the comments of the reviewers although they were only occasionally followed.

Visual Thinking in the Development of Dalton's Atomic Theory

Paul Thagard and Susan Hardy
Cognitive Science Laboratory
Princeton University
221 Nassau St., Princeton, NJ 08542

Abstract

This paper describes the role of visual thinking in the development of the atomic theory of the constitution of gases that John Dalton constructed around 1805. After briefly reviewing the psychological evidence for the existence of visual thinking, we identify several stages in Dalton's thought processes where visual thinking was important. Dalton used diagrams to work out the consequences of hypotheses he was developing and he employed diagrams to explain his hypotheses to others, sometimes using visual analogies based on mental images. In addition, we conjecture that images may also have played a role in forming his hypotheses. A knowledge representation scheme for computational imagery developed by Janice Glasgow and her colleagues can be used to help understand some of the structures and processes underlying Dalton's thinking.

1 Introduction

Imagine the letter 'B'. Rotate it 90 degrees to the left. Put a triangle below it having the same width and pointing down. Remove the horizontal line [Finke, Pinker and Farah 1989, p. 62].

Most people recognize the resulting emergent pattern as a heart. Such visual thinking is common in everyday life and also plays a role in scientific thinking, if we can believe the reports and notebooks of such scientists as Einstein. Yet, with a few exceptions listed below, visual thinking has been little studied by researchers in artificial intelligence. The normal tools of AI such as list processing languages are on the surface not well suited to describe the structures and processes of thinking with pictorial representations. Such representations can be external to the thinker as diagrams are, or can be internal to the thinker's imagination. In either case, the use of pictorial representations can facilitate processes that would be difficult to duplicate using propositional representations.

Numerous psychological experiments have been conducted that suggest that visual imagery can contribute to human cognition. Paivio [1986] showed that recall of words can be affected by imagery instructions. Shepard and his collaborators found that mental rotation of imaged objects behaves as one would expect if visual images were actually being rotated [Shepard and Cooper 1982]. Kosslyn and his colleagues systematically showed that people scan images in ways that suggest they are using pictorial representations [Kosslyn 1980]. All these results are open to challenges that people do not really use picture-like mental images, but instead merely use tacit propositional knowledge to mimic the use of visual imagery [Pylyshyn 1984]. But impressive evidence that visual imagery is closely related to visual perception comes from neuropsychological studies involving brain activity and selective effects of brain damage [Farah 1988]. Sloman [1978] and Larkin and Simon [1987] have provided compelling arguments for the computational power of pictorial representations. Substantial anecdotal evidence for the importance of visual thinking in science has been compiled by historians, philosophers, and psychologists who have noted the role of imagistic and diagrammatic cognition in such scientists as Bohr, Boltzmann, Einstein, Faraday, Heisenberg, Helmholtz, Herschel, Kekulé, Maxwell, Poincaré, Tesla, Watson, and Watt [Miller 1984; Nersessian 1992; Shepard 1988].

This research was supported by contract MDA903-89-K-0179 from the Basic Research Office of the U.S. Army Research Institute for the Behavioral and Social Sciences. We thank Christopher Tanner for programming assistance and Aaron Sloman for a pointer in this direction. Greg Nelson and David Gochfeld also helped. As of July, 1992, Thagard's address is Department of Philosophy, University of Waterloo.

2 Dalton's Atomic Theory

The use of visual thinking by John Dalton does not, however, seem to have been noted or analyzed.[1] Dalton, who was the first to publish a table of atomic weights, is viewed as one of the founders of modern chemistry. In 1793 he proposed that atmospheric air was a mixture of gases rather than a compound, and a series of experimental and theoretical investigations led him by around 1803 to realize that gases consist of atoms and that elements such as hydrogen and oxygen combine in multiple proportions. Early biographers of Dalton maintained that he first inductively discovered the law of multiple proportions and then formed the hypothesis of atomic structure to explain it. More careful examination of Dalton's papers before they were destroyed in a World War II air raid on Manchester uncovered, however, that Dalton's thought was more theory-driven. Reflection on the physical properties of the atmosphere and other gases led to a crude atomic theory which guided him to the discovery of regularities in chemical combination [Roscoe and Harden 1896, pp. 50-51].

Dalton himself gave a historical sketch of the development of his views on chemical elements in a lecture given in 1810; this lecture was preserved by Roscoe and Harden [1896, pp. 13-18]. Dalton says that his meteorological observations and speculations about the nature of the atmosphere led him to wonder how an atmosphere consisting of different elastic fluids (gases) could constitute a homogeneous mass. He took from Newton's *Principia* the idea that gases consist of small particles which repel each other, but could not make this consistent with the knowledge that the atmosphere contains at least three gases (oxygen, water, and azote, i.e. nitrogen) with different specific gravities. He began working with a theory of chemical affinity according to which the atmosphere did not subside into strata with heavier gases at the bottom because of a slight affinity between particles of gases of different kinds. But visual reasoning showed this view to be flawed:

> In order to reconcile or rather adapt this chemical theory of the atmosphere to the Newtonian doctrine of repulsive atoms or particles, I set to work to combine my atoms upon paper. I took an atom of water, another of oxygen, and another of azote, brought them together, and threw around them an atmosphere of heat, as per diagram; I repeated the operation, but soon found that the water particles were exhausted (for they make but a small part of the atmosphere). I next combined my atoms of oxygen and azote, one to one; but I found in time my oxygen failed; I then threw out all the remaining particles of azote into the mixture, and began to consider how the general equilibrium was to be obtained. [Roscoe and Harden 1896, pp. 14-15]

What Dalton was doing here is hard to follow, largely because the diagram he refers to has not survived. But it is clear that Dalton found great utility in using a diagram to work out the consequences of the chemical affinity view, finding spatial reasons for the incoherence of that view with Newtonian ideas about particles:[2]

> My triple compounds of water, oxygen, and azote were wonderfully inclined, by their superior gravity, to descend and take the lowest place; the double compounds of oxygen and azote affected to take a middle station, and the azote was inclined to swim at the top. I remedied this defect by lengthening the wings of my heavy particles, that is, by throwing more heat around them, by means of which I could make them float in any part of the vessel, but this change unfortunately made the whole mixture of the same specific gravity as azotic gas -- this circumstance could not for a moment be tolerated. In short, I was obliged to abandon the hypothesis of the chemical constitution of the atmosphere altogether, as irreconcilable to the phenomena. [Roscoe and Harden 1896, p. 15]

Here we see Dalton altering his diagram to attempt to reconcile chemical and Newtonian views, modifying the relation of particles and heat, but finding that the reconciliation fails. The visual representation seems to be playing a role in hypothesis formation when he talks of "lengthening the wings" of some of the particles rather than of conjecturing that the particles had longer wings. We will provide a computational interpretation of Dalton's visual reasoning below.

Dalton reported that in 1801 he hit upon a new hypothesis, that the atoms of one kind repel only atoms of their own kind, which explained why gases diffused through each other, but this contradicted the finding that diffusion is a slow process. Then came his big breakthrough:

> Upon reconsidering this subject, it occurred to me that I had never contemplated the effect of *difference of size* in the particles of elastic

[1] Langley, Simon, Zytkow and Bradshaw [1987] describe a program called DALTON that infers the structure of chemical compounds from reactions involving them. DALTON models the application of a version of Dalton's atomic theory, not its development, and it ignores visual thinking.

[2] Thagard [1992] categorizes scientific discoveries as data-driven, explanation-driven, and coherence-driven; Dalton here evidently falls into the third category.

> fluids. By *size* I mean the hard particle at the centre and the atmosphere of heat taken together. If, for instance, there be not exactly the same *number* of atoms of oxygen in a given volume of airs, as of azote in the same volume, then the *sizes* of the particles of oxygen must be different from those of azote. And if the *sizes* be different, then on the supposition that the repulsive power is heat, no equilibrium can be established by particles of unequal sizes pressing against each other. (See Diagram.) [Roscoe and Harden 1896, p. 16]

Unfortunately, the diagram referred to in this passage is also lost. But Dalton's reference to it suggests the pictorial character of his hypothesis that atoms of different kinds have different sizes. This hypothesis initiated according to Dalton a train of investigation for determining the number and weight of all chemical elements that enter into combination with each other.

In 1808, Dalton published a treatise, *A New System of Chemical Philosophy*, presenting his new theory and related material. This book included several plates that help enormously to convey how Dalton conceived of the structure of gases. In our figure 1 we reproduce Dalton's Plate 7 [Dalton 1808, vol. 1, p. 565]. His explanation of the plate is as follows:

> PLATE 7. Fig. 1, 2, and 3 represent profile views of the disposition and arrangement of particles constituting elastic fluids, both simple and compound but not mixed; it would be difficult to convey an adequate idea of the last case, agreeable to the principles maintained, page 190. - The principle may, however, be elucidated by the succeeding figures.
>
> Fig. 4 is the representation of 4 particles of azote with their elastic atmospheres, marked by rays (of heat) emanating from the solid central atom; these rays being exactly alike in all the 4 particles, can meet each other, and maintain an equilibrium.
>
> Fig. 5 represents 2 atoms of hydrogen drawn in due proportion to those of azote, and coming in contact with them; it is obvious that the atoms of hydrogen can apply one to the other with facility, but can not apply to those of azote, by reason of the rays not meeting each other in like circumstances; hence, the cause of the intestine motion which takes place on the mixture of elastic fluids, till the exterior particles come to press on something solid. [Dalton 1808, vol. 1, p. 548].

Here we see Dalton naturally presenting his ideas using diagrams, crude versions of which can be found in manuscripts written earlier [Roscoe and Harden 1896, plates 5 and 6]. The combination of Dalton's Fig. 4 and Fig. 5 provide much of his explanation of why gases stay mixed: the heat lines of atoms of the same kind meet, but the heat lines of azote atoms do not meet the heat lines of hydrogen atoms. Drawing these pictures and seeing how atom-heat combinations of different sizes do not have rays meeting could well have been the main source of Dalton's hypothesis.[3]

Not all of Dalton's visual thinking involved diagrams. He also used visual analogies to convey the spatial structures he was hypothesizing. He wrote that "every atom has an atmosphere of heat around it, in the same manner as the earth or any other planet has its atmosphere of air surrounding it, which cannot certainly be said to be held by chemical affinity, but by a species of attraction of a very different kind." [Roscoe and Harden 1896, p. 71] Here he maps an atom to the planet earth and the heat surrounding the atom to the earth's atmosphere, generating understanding of his proposed structure by comparison to something familiar. Similarly, he conveys the three-dimensional spatial organization of particles with a visual analogy:

> When we contemplate upon the disposition of the globular particles in a volume of pure elastic fluid, we perceive it must be analogous to that of a square pile of shot; the particles must be disposed into horizontal strata, each four particles forming a square: in a superior stratum each particle rests upon four particles below. [Dalton 1808, vol. 1, p. 189].

Our claim that these analogies are inherently visual will be defended in the course of our computational analysis of them.

3 A Knowledge Representation Scheme

Dalton's use of visual thinking can in part be understood using the knowledge representation scheme for computational imagery that Glasgow and her colleagues have been developing [Glasgow 1990; Glasgow and Papadias in press; Papadias and Glasgow 1991].[4] After summarizing that approach, we will apply it to analyze aspects of Dalton's work.

---

[3] In modern chemistry, explanation of why gases stay mixed is provided by the kinetic theory of gases which was developed several decades after Dalton. Dalton, like Lavoisier and other predecessors, thought heat was a substance rather than a form of energy.

[4] A terminological note: Glasgow follows Kosslyn in distinguishing visual information about what an object looks like from spatial information about where it is located in relation to other objects. Our use of the term "visual" involves both sorts of information.

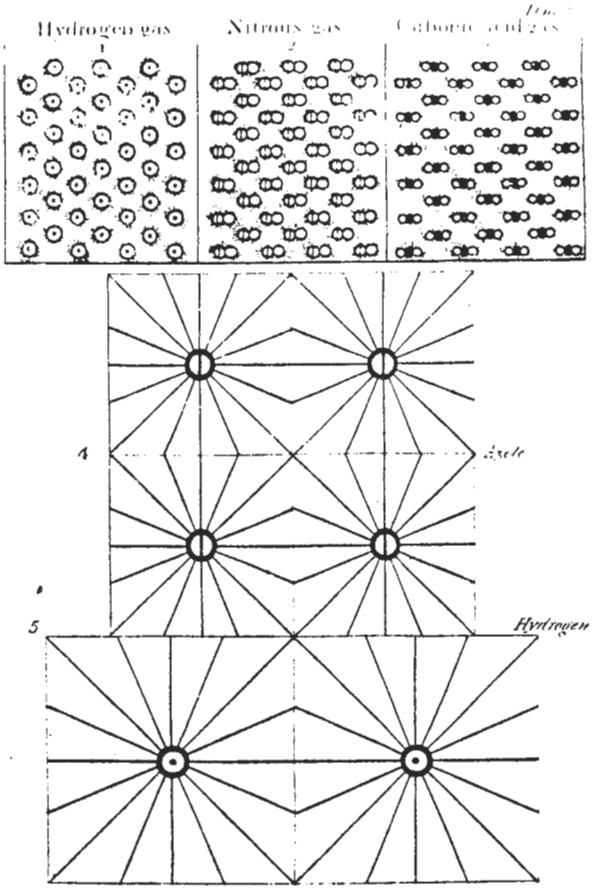


Figure 1. Dalton's diagrams.

In the earlier computational model of Kosslyn [1980], quasi-pictorial images were represented by a configuration of points in a matrix; an image is displayed by selectively filling in cells of the matrix. An image, then, is construed as a two-dimensional array, with each entry like a pixel that is either on or off. Glasgow's scheme is more complex in two key respects. First, it takes images to be inherently three-dimensional, although two-dimensional projects can also be handled as a special case. Greater dimensionality obviously makes possible representation of more complex images such as those required for mental rotation, as well as for some of Dalton's thinking as we will see below. Second, the entries in the three-dimensional arrays can be encoded hierarchically, in that each entry is represented symbolically by a entry that can have a subimage. For example, a house could be represented by the array shown in Figure 2, with each symbolic entry such as "window" providing a pointer to another array. In sum, Glasgow's representational scheme takes images to be three-dimensional symbolic hierarchical arrays. Numerous important visual operations can be defined on Glasgow's arrays, including constructing symbolic arrays from propositional representations, comparing images using array information, and moving and rotating images.

roof	roof	roof
window		
	door	

Figure 2: Array representation of a house.

Other researchers have developed models of visual thinking. Funt's [1980] model of diagrammatic problem solving used two-dimensional arrays with simple entries, like Kosslyn's. Shrager [1990] has constructed a system for simulating the perceived operation of a laser using two-dimensional arrays whose entries are label lists. His entries thus seem to be more complex than Kosslyn's, although from his description they do not seem to have the hierarchical character of Glasgow's scheme, nor are they three-dimensional. Chandrasekaran and Narayanan [1990] present a scheme that is similar to Glasgow's in allowing a hierarchy of descriptions, but no array organization is used. We shall see that the three-dimensional and hierarchical character of Glasgow's representations are both useful for understanding Dalton's visual thinking.

4 Computational Analysis of Dalton

We saw in figure 1 (Dalton's figures 4 and 5) that Dalton used rays to represent how atoms are surrounded by heat. In Glasgow's scheme, this corresponds to having an entry for each atom surrounded in three dimensions by entries for heat, as in our figure 3a, which shows only the two dimensional projection. We have added to Glasgow's primitive operations the operation SURROUND, which places a new entry in every place adjacent in all three dimensions to the place occupied by a given entry.[5] Dalton's attempt to "lengthen the wings" of heavy particles by adding more heat can be modelled using another application of SURROUND, to produce the result shown in two dimensions in figure 3b. When Dalton undertakes the task of combining these atom + heat clusters on paper, the hierarchical nature of Glasgow's representational scheme becomes useful, since the clusters can be manipulated as wholes. We thus gain the advantage found in object-oriented drawing programs such as MacDraw, in contrast to bitmap-oriented programs such as MacPaint. Figure 3c shows how a picture can be drawn of the constitution of the atmosphere using higher-level symbolic elements, each of which represents an array of the sort shown in figures 3a and 3b. The frequencies of the different entries roughly represent Dalton's knowledge that the atmosphere contains more nitrogen than oxygen, and more oxygen than water vapor.

[5] Glasgow and her colleagues have elegantly implement-

```
HEAT HEAT HEAT

HEAT ATOM HEAT

HEAT HEAT HEAT
```

Figure 3a

```
HEAT HEAT HEAT HEAT HEAT

HEAT HEAT HEAT HEAT HEAT

HEAT HEAT ATOM HEAT HEAT

HEAT HEAT HEAT HEAT HEAT

HEAT HEAT HEAT HEAT HEAT
```

Figure 3b

```
OXYGEN+HEAT   AZOTE+HEAT   AZOTE+HEAT

AZOTE+HEAT    OXYGEN+HEAT  AZOTE+HEAT

AZOTE+HEAT    AZOTE+HEAT   WATER+HEAT
```

Figure 3c

Figure 3. Array representations for Dalton.

Dalton's big breakthrough came by his realization that atoms can have different sizes. Presumably this can be represented in the Glasgow scheme by having entries spread over more than one array location. There is a possible ambiguity here, however, if we want to distinguish between BALL BALL BALL, which is three balls in a row, and ROOF ROOF ROOF, which is one continuous roof. Similarly, we want OXYGEN OXYGEN to represent a large oxygen atom, not two. We could then use something like figure 3c to convey the overall structure of a gas, but have sub-images of different sizes, so that oxygen clusters are bigger than nitrogen clusters.

In section 2, we quoted Dalton's use of two analogies that are naturally understood using visual representations. The simplest is that an atom is surrounded by an atmosphere of heat just as the earth is surrounded by an atmosphere of air. Current AI programs for analogical mapping such as ACME [Holyoak and Thagard 1989; Thagard, Cohen, and Holyoak 1989] and SME [Falkenhainer, Forbus, and Gentner 1989] take as input predicate calculus representation. Thus ACME might be given the input:

```
Source: (PLANET (EARTH))
    (ATMOSPHERE (AIR))
    (SURROUND (AIR EARTH))
Target: (ATOM (ATOM-OBJ))
    (HEAT (HEAT-OBJ))
    (SURROUND (HEAT-OBJ ATOM-OBJ))
```

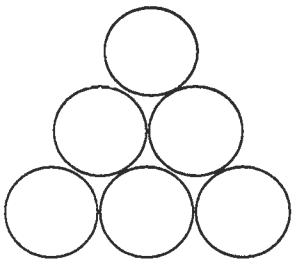
ACME would easily map the source to the target by pairing up SURROUND in the source with SURROUND in the target and sorting out all the other relations needed to maintain isomorphism, such as mapping EARTH to ATOM-OBJ. If, however, spatial structure is represented by three dimensional arrays, the mapping process is very different. The relation *surround* is not explicitly represented at all, but only implicitly by the fact that the array location for the entry EARTH has the entry AIR in all adjacent locations. Similarly, the array location for ATOM-OBJ has the the entry HEAT-OBJ in all adjacent locations. The mapping of EARTH to ATOM-OBJ is virtually immediate once the two arrays are compared with each other. Unlike ACME, which at least considers the mapping of AIR to ATOM-OBJ before eventually rejecting it as inferior, the visual mapping could line things up much more directly, with mapping coming automatically from superimposition of one array onto the other. Of course, this presupposes that the two three-dimensional arrays are the same size.

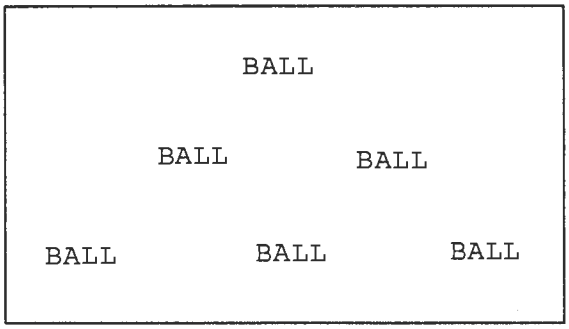
Visual representations are also very useful for creating analogies, as in Dalton's comparison of the structure of the atmosphere with a pile of shot. It is easy to construct a mental image of a pile of cannon balls with one ball nesting on four below which nest on nine below, and then transform this into a picture of the atmosphere consisting of atoms surrounded by heat similarly nesting. Figure 4 shows several views of the pile of balls, the first from the front and the other three considering each layer only. To the right of the pictures of balls is a diagram that shows how the picture can be represented in layers of a three-dimensional array. The representation of the pile of balls is not just the various slices shown, but the whole array which encapsulates a very large amount of spatial information. This encapsulation makes creating a visual analog trivial: all we have to do to produce a representation of the structure of the atmosphere is to replace each entry of BALL with an entry of ATOM+HEAT. Once again, the hierarchical nature of the representation scheme is useful, since it allows us to substitute a complex of atom and heat rather than just atom.

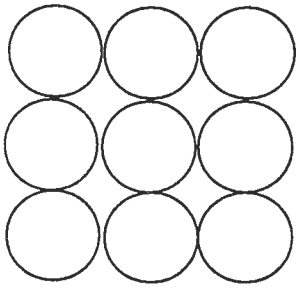
Contrast what would be involved in representing and transferring this information using ACME or SME. First, we would need a large number of propositions to capture

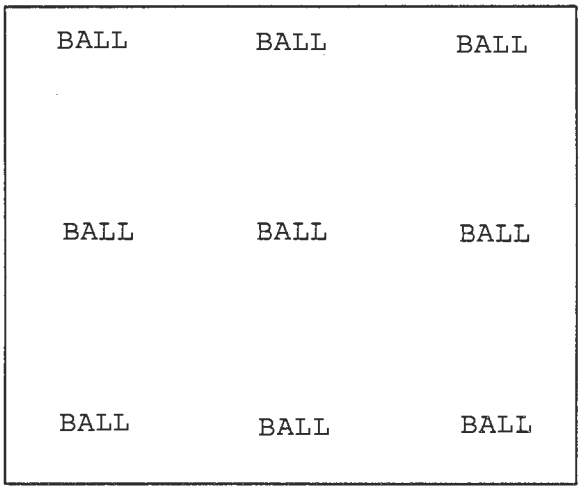
---

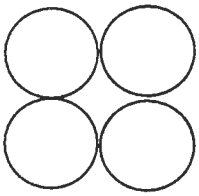
ed her scheme in the array processing language Nial. We have developed a Common LISP implementation of the spatial part of her scheme, encoding three-dimensional arrays as lists of lists of lists.

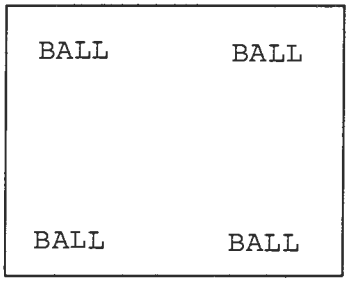












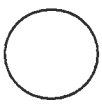


Figure 4. Array representation of a pile of balls.

the structure of the pile of balls, enumerating the balls from the bottom up and front to back and left to right:

(BALL (BALL1)) (BALL (BALL2) (BALL (BALL3))
(BALL (BALL4)) (BALL (BALL5))(BALL (BALL6))
(BALL (BALL7) (BALL (BALL8)) (BALL (BALL9))
(BALL (BALL10)) (BALL (BALL11))
(BALL (BALL12)) (BALL (BALL13))
(FRONT-OF (BALL1 BALL2)) (FRONT-OF (BALL2 BALL3))
(FRONT-OF (BALL4 BALL5)) (FRONT-OF (BALL5 BALL6))
(FRONT-OF (BALL7 BALL8)) (FRONT-OF (BALL8 BALL9))
(FRONT-OF (BALL10 BALL11)) (FRONT-OF (BALL12 BALL13))
(LEFT-OF (BALL1 BALL4)) (LEFT-OF (BALL4 BALL7))
(LEFT-OF (BALL2 BALL5)) (LEFT-OF (BALL5 BALL8))
(LEFT-OF (BALL3 BALL6)) (LEFT-OF (BALL6 BALL9))
(LEFT-OF (BALL10 BALL12)) (LEFT-OF (BALL11 BALL13))
(TOP-OF (BALL14 BALL10 BALL11 BALL12 BALL13))
(TOP-OF (BALL10 BALL1 BALL2 BALL4 BALL5))
(TOP-OF (BALL11 BALL2 BALL3 BALL5 BALL6))
(TOP-OF (BALL12 BALL4 BALL5 BALL7 BALL8))
(TOP-OF (BALL13 BALL5 BALL6 BALL8 BALL9))

Constructing this encoding would be very difficult without first making a diagram! Moreover, the process of mapping it to an isomorphic structure with atoms or atom-heat clusters substituted for balls will be computationally expensive, whereas a visual analogical mapper that presupposes primitive operations on three-dimensional arrays makes the comparison trivial. The implementation of these operations in a list or array processing languages is not so trivial, but if one assumes that the human information processing system comes already with an evolved massively parallel apparatus for visual comparisons that can be taken over and applied without additional calculations, then the advantages of doing at least some analogical mapping visually are apparent. It would be useful to reinterpret as visual many of the standard examples that have been used in discussions of analogical mapping, for example the comparison of the Bohr model of the atom with the solar system [Gentner 1983], the solving of Duncker's ray problem using a military analogy [Gick and Holyoak 1980], and the use of an analogy with water waves to discover the wave theory of sound [Thagard 1988]. Visual representations may also be useful in providing a new approach to analog retrieval [Thagard, Holyoak, Nelson, and Gochfeld 1990].

We have therefore extended our Common LISP implementation of parts of Glasgow's scheme to constitute a program called VAMP.1, for Visual Analogical Mapping Program, version 1.[6] Given two arrays, VAMP.1 can do simple analogical mapping, putting the elements of the two arrays in correspondence with each other. VAMP.1 first checks to see if the arrays are the same size. If not, it scales them up to the size of the least common multiple of their sizes. For example, to compare a 4x4x4 array and a 6x6x6 array, VAMP.1 converts both arrays to 12x12x12 arrays. When both arrays are equal in size, VAMP.1 superimposes them and gives a list of all parts which are in corresponding cells. Thus VAMP.1 can quickly map a 3-dimensional representation of an atmosphere consisting of Dalton atoms onto a pile of shot and infer the appropriate correspondence between particular atoms and particular balls. In contrast, when ACME is given the same information expressed in predicate calculus, it creates a constraint network of more than 3500 units and exhausts the memory on our Sparcstation 2.

Thus several aspects of the use of diagrams and visual analogies by Dalton can be understood in terms of Glasgow's representational scheme for computational imagery. Of course, that scheme does not tell the whole story. More finely tuned methods are needed to represent such structural relations as the angles at which the cannon balls touch each other. We would not want, however, to go right down to the pixel level, because then we would lose the ability to manipulate objects and different levels of detail that is so natural in Glasgow's scheme. We have seen that visual thinking was clearly important in the development and exposition of Dalton's atomic theory, and that aspects of this development can fruitfully be understood in terms of three-dimensional hierarchical symbolic arrays.

References

Chandrasekaran, B., and Narayanan, N. [1990]. Integrating imagery and visual representations. *Proceedings of the 12th Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum, 670-677.

Dalton, J. [1808]. *A new system of chemical philosophy*. London: Bickerstaff.

Falkenhainer, B., Forbus, K., and Gentner, D. [1989] The structure-mapping engine: Algorithms and examples. *Artificial Intelligence*, 41, 1-63.

Farah, M. [1988]. Is visual imagery really visual? Overlooked evidence from neuropsychology. *Psychological Review, 95, 307-317.*

---

[6] Very recently, we have completed VAMP.2, which draws on ideas from connectionism and Minsky's Society of Mind to implement the hierarchical and 3-D aspects of visual representation without using arrays. We hope eventually to integrate VAMP.2 with CARE, a new system that combines analogical and rule-based reasoning using parallel constraint satisfaction in localist networks [Nelson, Thagard, and Hardy 1992]. The new system would be capable of doing analogical mapping simultaneously in two modes, using VAMP.2's visual representations and CARE's ACME-like propositional ones. Our conjecture is that the two kinds of mapping programs should be able to work cooperatively to find correspondences between analogs more quickly and comprehensively than either could do alone.

Finke, R., Pinker, S., and Farah, M. [1989]. Reinterpreting visual patterns in mental imagery. *Cognitive Science, 13, 51-78.*

Funt, B. [1980]. Problem solving with diagrammatic representations. *Artificial Intelligence, 13,* 201-230.

Gentner, D. [1983]. Structure-mapping: A theoretical framework for analogy. *Cognitive Science, 7,* 155-170.

Gick, M., and Holyoak, K. [1980]. Analogical problem solving. *Cognitive Psychology, 12,* 306- 355.

Glasgow, J. [1990]. Imagery and classification. *Proceedings of the 1st ASIS SIG/CR Classification Research Workshop.* Toronto.

Glasgow, J., and Papadias, D. [in press]. Computational imagery. *Cognitive Science.*

Holyoak, K. and Thagard, P. [1989] Analogical mapping by constraint satisfaction. *Cognitive Science, 13,* 295-355.

Kosslyn, S. [1980]. *Image and mind.* Cambridge: Harvard University Press.

Larkin, J., and Simon, H. [1987]. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science, 11:* 65-100.

Miller, A. I. [1984]. *Imagery in scientific thought: Creating twentieth century physics.* Boston: Birkhauser.

Nelson, G., Thagard, P., and Hardy, S. [in press]. Integrating analogies with rules and explanations. In J. Barnden and K. Holyoak (Eds.), *Advances in Connectionism,* vol. 2, *Analogical Connections,* Norwood, NJ: Ablex.

Nersessian, N. [1992]. How do scientists think? In R. Giere (Ed.), *Cognitive Models of Science, Minnesota Studies in the Philosophy of Science,* vol. 15. Minneapolis: University of Minnesota Press, in press.

Papadias, D., and Glasgow, J. [1991]. A knowledge representation scheme for computational imagery. *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society,* Hillsdale, NJ: Erlbaum, 49-54..

Pylyshyn, Z. [1984]. *Computation and cognition: Toward a foundation for cognitive science.* Cambridge, MA: MIT Press.

Roscoe, H., and Harden, A. [1896]. *A new view of the origin of Dalton's atomic theory.* London: Macmillan. Reprint 1970, Johnson Reprint Corporation, New York.

Shepard, R. [1988]. The imagination of the scientist. In K. Egan and D. Nadaner (Eds.), *Imagination and Education.* New York: Teachers College Press, 153-185.

Shepard, R., and Cooper, L. [1982]. *Mental images and their transformations.* Cambridge, MA: MIT Press.

Shrager, J. [1990]. Commonsense perception and the psychology of theory formation. In J. Shrager and P. Langley (Eds.), *Computational models of discovery and theory formation.* San Mateo, CA: Morgan Kaufman. 437-470.

Sloman, A. [1978]. *The computer revolution in philosophy.* Atlantic Highlands: Humanities Press.

Thagard, P. [1988]. *Computational philosophy of science.* Cambridge, MA: MIT Press/Bradford Books.

Thagard, P. [1992]. *Conceptual revolutions.* Princeton University Press.

Thagard, P., Cohen, D., and Holyoak, K. [1989] Chemical analogies: Two kinds of explanation. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.* San Mateo: Morgan Kaufmann, 819-824.

Thagard, P., Holyoak, K., Nelson, G., and Gochfeld, D. [1990]. Analog retrieval by constraint satisfaction. *Artificial Intelligence, 46,* 259-310.

Efficient Algorithms for Identifying Relevant Features*

Hussein Almuallim and **Thomas G. Dietterich**
303 Dearborn Hall
Department of Computer Science
Oregon State University
Corvallis, OR 97331-3202
U. S. A.
almualh@cs.orst.edu
tgd@cs.orst.edu

Abstract

This paper describes efficient methods for exact and approximate implementation of the MIN-FEATURES bias, which prefers consistent hypotheses definable over as few features as possible. This bias is useful for learning domains where many irrelevant features are present in the training data.

We first introduce FOCUS-2, a new algorithm that exactly implements the MIN-FEATURES bias. This algorithm is empirically shown to be substantially faster than the FOCUS algorithm previously given in [Almuallim and Dietterich, 1991]. We then introduce the Mutual-Information-Greedy, Simple-Greedy and Weighted-Greedy algorithms, which apply efficient heuristics for approximating the MIN-FEATURES bias. These algorithms employ greedy heuristics that trade optimality for computational efficiency. Experimental studies show that the learning performance of ID3 is greatly improved when these algorithms are used to preprocess the training data by eliminating the irrelevant features from ID3's consideration. In particular, the Weighted-Greedy algorithm provides an excellent and efficient approximation of the MIN-FEATURES bias.

1 Introduction

In many inductive learning applications, one has to deal with training data that contain many features that are irrelevant to the target concept being learned. In these domains, an appropriate bias is the MIN-FEATURES bias, which prefers any consistent hypothesis definable over as few features as possible. Previous work [Almuallim and Dietterich, 1991] showed that this simple bias is strong enough to yield polynomial sample complexity. The same study showed that—contrary to expectations—the performance of conventional inductive learning algorithms such as ID3 [Quinlan, 1986] and FRINGE [Pagallo and Haussler, 1990] is seriously reduced by the presence of irrelevant features. These results suggested that one should not rely on these algorithms to filter out irrelevant features. Instead, some technique should be employed to eliminate irrelevant features and focus the learning algorithm on the relevant ones.

Given a set of training examples of an unknown target concept, the task for any algorithm implementing the MIN-FEATURES bias is to find the smallest subset of the given features that permits a consistent hypothesis to be defined. In previous work [Almuallim and Dietterich 91], an algorithm called FOCUS was presented that exactly implements the MIN-FEATURES bias. However, the worst-case running time of this algorithm is exponential in the number of relevant features. The goal of this paper is to describe more efficient algorithms for exact and approximate implementation of the MIN-FEATURES bias.

Specifically, the paper first introduces FOCUS-2, a new algorithm that exactly implements the MIN-FEATURES bias. This algorithm is empirically shown to be substantially faster than FOCUS. We then introduce the Mutual-Information-Greedy, Simple-Greedy and Weighted-Greedy algorithms, which apply efficient heuristics for approximating the MIN-FEATURES bias. Unlike FOCUS-2, these algorithms employ *greedy* heuristics that trade optimality for computational efficiency. Experimental studies show that the learning performance of ID3 [Quinlan, 1986] is greatly improved when these algorithms are used to preprocess the training data by eliminating the irrelevant features from ID3's consideration. In particular, the Weighted-Greedy algorithm provides an excellent and efficient approximation of the MIN-FEATURES bias.

The task of selecting a subset of the available features that meets a given criterion has long been known in the field of pattern recognition as the problem of "feature selection" or "dimensionality reduction." However, most of the work in this area seeks to enhance the computational efficiency of particular classifiers while leaving their accuracy unaffected, whereas the goal of this paper is to improve the accuracy of the classifier by selecting the minimal number of features.

The typical case studied in pattern recognition involves a classifier that is capable of performing quite

*The authors gratefully acknowledge the support of the NSF under grant number IRI-86-57316. Hussein Almuallim was supported by a scholarship from the University of Petroleum and Minerals, Dhahran, Saudi Arabia.

well without feature selection (i.e., using all of the available features). However, for ease of hardware implementation and speed of processing, it is necessary to reduce the number of features considered by the classifier. Generally, the classifiers studied in pattern recognition have the so-called monotonicity property ([Narendra and Fukunaga, 1977]) that as the number of features is reduced, the accuracy decreases. The goal of feature selection is to eliminate as many features as possible without significantly degrading performance.

Most feature selection criteria in pattern recognition are defined with respect to a specific classifier or group of classifiers. For example, [Kittler, 1980] show methods for selecting a small subset of features that optimizes the expected error of the nearest neighbor classifier. Similar work has addressed feature selection for the Box classifier [Ichino and Sklansky, 1984a], the linear classifier [Ichino and Sklansky, 1984b] and the Bayes classifier [Queiros and Gelsma, 1984]. Other work (aimed at removing feature redundancy when features are highly correlated) is based on performing a principal components analysis to find a reduced set of new uncorrelated features defined by *combining* the original features using the eigenvectors [Morgera, 1986; Mucciardi and Gose, 1971]. To our knowledge, the problem of finding the smallest subset of Boolean features that is sufficient to construct a consistent hypothesis (regardless of the *form* of the hypothesis)—which is the topic of this paper—has not been addressed.

2 Preliminaries

Let $\{x_1, x_2, \cdots x_n\}$ be a set of n Boolean features, and let U_n denote the set of all possible assignments to these features. A *concept* c is a subset of U_n (i.e., all positive instances of c). An *example* for a concept c is a pair $\langle X, class\rangle$, where $X \in U_n$ and *class* is $+$ if $X \in c$ and $-$ otherwise. A *sample* is a set of examples drawn at random from U_n.

Given a training sample and a set of features Q, a *sufficiency test* is a procedure for checking whether Q is sufficient to form a consistent hypothesis. The sufficiency test can be implemented simply by checking whether the sample contains a pair $\langle X_1, +\rangle$ and $\langle X_2, -\rangle$ of positive and negative examples such that X_1 and X_2 have the same values for all the features in Q. If such a pair appears, then Q cannot discriminate all of the positive examples from all of the negative examples. In general, Q is a sufficient set if and only if no such a pair appears in the training sample.

For a pair of examples $\langle X_1, +\rangle$ and $\langle X_2, -\rangle$, we define a *conflict* generated from this pair as an n-bit vector $a = \langle a_1 a_2 \cdots a_n\rangle$ where $a_i = 1$ if X_1 and X_2 have different values for the feature x_i and 0 otherwise. We will say that a is *explained* by x_i if and only if $a_i = 1$. Using this terminology, a set Q of features is sufficient to construct a hypothesis consistent with a given training sample if and only if every conflict generated from the sample is explained by some feature in Q.

Example: Let the training sample be

$\langle 010100, +\rangle$ $\quad$ $\langle 011000, -\rangle$
$\langle 110010, +\rangle$ $\quad$ $\langle 101001, -\rangle$
$\langle 101111, +\rangle$ $\quad$ $\langle 100101, -\rangle$

Then, the set of all conflicts generated from this sample is

$a_1 = \langle 001100\rangle$ $\quad$ $a_4 = \langle 101010\rangle$ $\quad$ $a_7 = \langle 110111\rangle$
$a_2 = \langle 111101\rangle$ $\quad$ $a_5 = \langle 011011\rangle$ $\quad$ $a_8 = \langle 000110\rangle$
$a_3 = \langle 110001\rangle$ $\quad$ $a_6 = \langle 010111\rangle$ $\quad$ $a_9 = \langle 001010\rangle$

The reader can check that the subset $\{x_1, x_3, x_4\}$ is sufficient to form a consistent hypothesis (e.g., $\bar{x}_1\bar{x}_3 \vee (\bar{x}_3 \oplus x_4)$), and that all subsets of cardinality less than 3 are insufficient. □.

Given a sufficient subset of features, it is easy to construct a consistent hypothesis. For example, the algorithm ID3 [Quinlan, 1986] can be applied to the training sample but restricted to consider only the features in the given subset. Hence, in the rest of this paper, finding a solution will be taken to mean identifying a subset of features sufficient to form a consistent hypothesis.

Algorithm FOCUS-2(*Sample*)
1. If all the examples in *Sample* have the same class, then return ϕ.
2. Let G be the set of all conflicts generated from *Sample*.
3. *Queue*= $\{M_{\phi,\phi}\}$.
   /* This is a first-in-first-out data structure. */
4. Repeat
   4.1. Pop the first element in *Queue*. Call it $M_{A,B}$.
   4.2. Let $OUT = A$.
   4.3. Let a be the conflict in G not explained by any of the features in A such that $|Z_a - B|$ is minimized, where Z_a is the set of features explaining a.
   4.4. For each $x \in Z_a - B$
      4.4.1. If Sufficient($A \bigcup \{x\}$), return ($A \bigcup \{x\}$).
      4.4.2. Insert $M_{A\bigcup\{x\},OUT}$ at the tail of *Queue*.
      4.4.3. $OUT = OUT \bigcup \{x\}$.

end FOCUS-2.

Figure 1: The FOCUS-2 learning algorithm.

3 Improving the FOCUS Algorithm

The FOCUS algorithm given in [Almuallim and Dietterich, 1991] works by trying all the subsets of features of increasing size until a sufficient set is encountered. In the example of the previous section, FOCUS tests the $\binom{6}{0} + \binom{6}{1} + \binom{6}{2} = 22$ subsets of features of size 0, 1 and 2, and some of the $\binom{6}{3} = 20$ subsets of size 3 before returning a solution. By doing so, FOCUS is not exploiting all the information given in the training sample. Consider, for instance, the conflict $a_1 = \langle 001100\rangle$. This conflict tells us that any sufficient set of features must contain x_3 or x_4 in order to explain the conflict. Hence, none of the sets $\{x_1\}, \{x_2\}, \{x_5\}, \{x_6\}, \{x_1, x_2\}, \{x_1, x_5\}, \{x_1, x_6\}$, $\{x_2, x_5\}$, $\{x_2, x_6\}$, $\{x_5, x_6\}$ can be solutions. Therefore, all of these sets can immediately be ruled out of the algorithm's consideration. Many other subsets can be similarly ruled out based on the other conflicts.

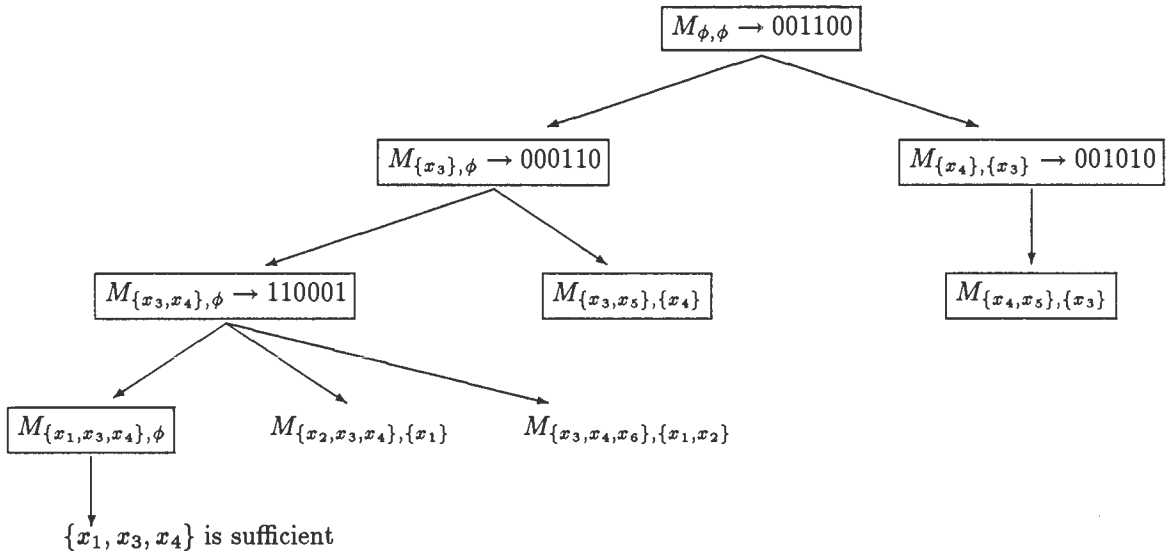


Figure 2: An example of FOCUS-2. Rectangles indicate where the sufficiency tests occurred.

Figure 1 shows the FOCUS-2 algorithm, which takes advantage of this observation. In this algorithm, we use a first-in-first-out queue in which each element denotes a subspace of the space of all feature subsets. Each element has the form $M_{A,B}$, which denotes the space of all feature subsets that include all of the features in the set A and none of the features in the set B. Formally,

$$M_{A,B} = \{T \mid T \supseteq A, T \bigcap B = \phi, T \subseteq \{x_1, x_2, \cdots, x_n\}\}.$$

For example, the set $M_{\phi,\phi}$ denotes all possible feature subsets, the set $M_{A,\phi}$ denotes all feature subsets that contain at least the features in A, and the set $M_{\phi,B}$ denotes all feature subsets that do not contain any features in B.

The main idea of FOCUS-2 is to keep in the queue only the *promising* portions of the space of feature subsets—i.e. those that *may* contain a solution. Initially, the queue contains only the element $M_{\phi,\phi}$ which represents the whole power set. In each iteration in Step 4, the space represented by the head of the queue is partitioned into disjoint subspaces, and those subspaces that cannot contain solutions are pruned from the search.

Consider again the conflict $a_1 = \langle 001100 \rangle$. Suppose the current space of possible feature subsets is $M_{\phi,\phi}$. We know that any sufficient feature subset must contain either x_3 or x_4. We can incorporate this knowledge into the search by refining $M_{\phi,\phi}$ into the two subspaces $M_{\{x_3\},\phi}$ (all feature subsets that contain x_3) and $M_{\{x_4\},\{x_3\}}$ (all feature subsets that contain x_4 and do not contain x_3). Note that the second subscript of M is used to keep the various subspaces disjoint. Clearly, conflicts with fewer 1's in them provide more constraint for the search than conflicts with more 1's. Hence, if the head of the queue is $M_{A,B}$, then the algorithm (Step 4.3) searches for a conflict a such that

- a is not explained by any of the features in A, and
- the number of 1's corresponding to features that are not in B is minimized.

The algorithm then incorporates a into the search.

In detail, here is how FOCUS-2 behaves on the example given in the previous section. As shown in Figure 2, the algorithm starts by processing $M_{\phi,\phi}$. The conflict $a_1 = \langle 001100 \rangle$ is selected in Step 4.3 and $M_{\phi,\phi}$ is replaced by $M_{\{x_3\},\phi}$ and $M_{\{x_4\},\{x_3\}}$. Next, for $M_{\{x_3\},\phi}$, the conflict $a_8 = \langle 000110 \rangle$ is selected, and $M_{\{x_3,x_4\},\phi}$ and $M_{\{x_3,x_5\},\{x_4\}}$ are added to the queue. $M_{\{x_4\},\{x_3\}}$ is then processed with $a_9 = \langle 001010 \rangle$ and $M_{\{x_4,x_5\},\{x_3\}}$ is inserted. Finally, when $M_{\{x_3,x_4\},\phi}$ is processed with $a_3 = 110001$, the algorithm terminates in Step 4.4.1 before adding $M_{\{x_1,x_3,x_4\},\phi}$ to the queue, since $\{x_1, x_3, x_4\}$ is a solution.

Using FOCUS-2, the number of sufficiency tests is only 7. By comparison, FOCUS must perform at least 23 sufficiency tests (to test each of the 22 subsets of size up to 2, and at least one of the 20 subsets of size 3). Because FOCUS-2 only prunes subspaces that cannot possibly explain all of the conflicts, it is sound and complete—it will not miss any sufficient feature subsets. Furthermore, because it considers the subspaces $M_{A,B}$ in order of increasing size of A, it is guaranteed to find a sufficient subset with the smallest possible size. Finally, of course, the number of sufficiency tests performed by FOCUS-2 will typically be much less (and certainly never more) than the number of tests performed by FOCUS.

4 Heuristics for the MIN-FEATURES bias

Exact implementation of the MIN-FEATURES bias in domains with large numbers of features can be computationally infeasible[1]. In such cases, one may be willing

[1]The reader may have already noticed the connection between the MIN-FEATURES bias and the Minimum-Set-

to employ efficient heuristics that provide good but not necessarily optimal solutions. In this section, we describe three such algorithms. Each of these algorithms implements an iterative procedure where in each iteration the feature that seems most promising is added to the partial solution. This continues until a sufficient set of features is found. The only difference between the three algorithms is the criterion used in selecting the best feature in each iteration.

In Figure 3, we give a sketch of each of our algorithms. More detailed and computationally efficient implementations can be found in [Almuallim, 1992]. In the following, we describe the selection criteria implemented by each algorithm.

The Mutual-Information-Greedy (MIG) Algorithm: For a given set of features Q, imagine that the training sample is partitioned into $2^{|Q|}$ groups such that the examples in each group have the same truth assignment to the features in Q. (One can think of this as a completely balanced decision tree with $2^{|Q|}$ leaves.) Let p_i and n_i denote the number of positive and negative examples in the i-th group, respectively. The *entropy* of Q is defined as

$$Entropy(Q) = -\sum_{i=0}^{2^{|Q|}-1} \frac{p_i + n_i}{|Sample|} \left[\frac{p_i}{p_i + n_i} \log_2 \frac{p_i}{p_i + n_i} + \frac{n_i}{p_i + n_i} \log_2 \frac{n_i}{p_i + n_i} \right]$$

with the convention that $a \log_2 a = 0$ when $a = 0$.

In the Mutual-Information-Greedy algorithm, the feature that leads to the minimum entropy when added to the current partial solution is selected as the best feature.

The Simple-Greedy (SG) Algorithm: This algorithm chooses each time the feature that explains the largest number of conflicts that are not yet explained. The conflicts that are explained by this feature are then removed from the set of conflicts. The process is repeated until all conflicts are removed.

The Weighted-Greedy (WG) Algorithm: In the Simple-Greedy algorithm, every conflict contributes a unit increment to the score of each feature that explains it. In the Weighted-Greedy algorithm, the increment instead depends on the total number of features that explain the conflict. The intuition is that if a feature uniquely explains a conflict, then that feature *must* be part of the solution set of features. If A_{x_i} is the set of conflicts explained by a feature x_i, then the score of x_i is computed as

$$score_{x_i} = \sum_{a \in A_{x_i}} \frac{1}{\#\text{ of features explaining } a \; - 1}$$

Cover problem, which is known to be NP-hard [Garey and Johnson, 1979]. However, note that we assume here the existence of a *small* set of features that forms a solution. This corresponds to restricting the Minimum-Set-Cover problem to instances that have small covers.

Algorithm: Mutual-Information-Greedy(*Sample*)
1. $Q = \phi$.
2. Repeat until $Entropy(Q) = 0$:
   2.1. For each feature x_i,
   let $score_{x_i} = Entropy(Q \bigcup \{x_i\})$.
   2.2. Let *best* be the feature with the lowest score.
   2.3. $Q = Q \bigcup \{ best \}$.

end Mutual-Information-Greedy

Algorithm: Simple-Greedy(*Sample*)
1. $Q = \phi$.
2. Let A be the set of all conflicts generated from *Sample*.
3. Repeat until A is empty:
   3.1. For each feature x_i, let $score_{x_i}$ = the number of conflicts explained by x_i.
   3.2. Let *best* be the feature with the highest score.
   3.3. $Q = Q \bigcup \{ best \}$.
   3.4. Remove from A all the conflicts explained by *best*.

end Simple-Greedy.

Algorithm: Weighted-Greedy(*Sample*)
1. $Q = \phi$.
2. Let A be the set of all conflicts generated from *Sample*.
3. Repeat until A is empty:
   3.1. For each feature x_i:
      3.1.1. A_{x_i} = the set of conflicts explained by x_i.
      3.1.2 $score_{x_i} = \sum_{a \in A_{x_i}} \frac{1}{\#\text{ of features explaining } a \; - 1}$.
   3.2. Let *best* be the feature with the highest score.
   3.3. $Q = Q \bigcup \{ best \}$.
   3.4. Remove from A all the conflicts explained by *best*.

end Weighted-Greedy.

Figure 3: Three heuristics for approximating the MIN-FEATURES bias.

Under this heuristic, when a feature x_i explains a conflict a, the contribution of a to the score of x_i is inversely proportional to the number of *other* features that explain a. If only a few other conflicts explain a then x_i receives high credit for explaining a. In the extreme case where a is exclusively explained by x_i, the score of x_i becomes ∞. This causes the feature to be included in the solution with certainty.

5 Experimental Results

5.1 Sample Complexity and Accuracy

In this subsection, we test the value of each of the heuristics of Section 4 for learning tasks where many irrelevant features are present. Note that these heuristics are not complete learning algorithms—rather they are preprocessors that provide us only with a set of features sufficient to construct a consistent hypothesis. To construct an actual hypothesis, we first filter the training examples

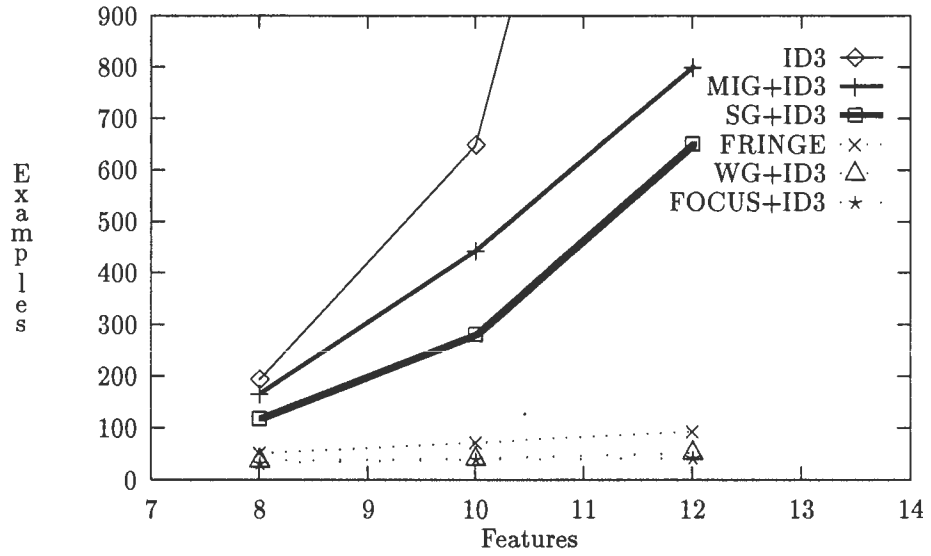


Figure 4: The number of examples needed for learning all the concepts with 3 relevant features out of 8, 10, and 12 available features. ID3 requires 2236 examples when the total number of features is 12.

to remove all features not selected during preprocessing. Then we give the filtered examples to ID3 to construct a decision tree. We will refer to the three algorithms as MIG+ID3, SG+ID3 and WG+ID3. Likewise, using FOCUS instead of these heuristics to find a sufficient subset of features will be denoted FOCUS+ID3.

For comparison, our experiments also include ID3 with no preprocessing in addition to FRINGE [Pagallo and Haussler, 1990]. Our version of ID3 performs no windowing or forward pruning and employs the information gain (mutual information) criterion to select features. FRINGE is terminated after at most 10 iterations.

The algorithms are evaluated through a set of experiments similar to those reported in [Almuallim and Dietterich, 1991]. Details of the experiments can be found in [Almuallim, 1992]. We report here two kinds of experiments. In the first experiment, we are interested in the *worst-case* performance over a class of concepts, where each concept is definable over at most p out of n features (and hence, the MIN-FEATURES bias is appropriate). As our measure of performance, we employ the sample complexity—the minimum number of training examples needed to ensure that every concept in the class can be learned in the PAC sense [Blumer *et al.*, 1987]. We estimate the sample complexity with respect to fixed learning parameters p, n, ϵ, and δ and with training samples drawn according to the uniform distribution.

In the second experiment, we are interested in the *average-case* performance of the algorithms. We randomly generated a collection of concepts that involve only a few features among many available ones. We then measured the accuracy rate of each algorithm while progressively increasing the size of the training sample—i.e. by plotting the learning curve for each of the concepts under consideration.

EXPERIMENT 1: Sample Complexity. The goal of this experiment is to estimate the minimum number of examples that enables each algorithm to PAC learn all the concepts of at most 3 relevant features out of n available features for $n = 8, 10$ and 12. To decide whether an algorithm L learns a concept c for sample size m, we generate 100,000 random samples of c of size m. We conclude that c is learned by L if and only if for at least 90% of these samples L returns a hypothesis that is at least 90% correct. Thus, the quantity measured here can be viewed as an empirical estimate of the *sample complexity* of each algorithm [Blumer *et al.*, 1987] for $\epsilon = \delta = 0.1$. To reduce the computational costs involved in this experiment, we exploited the fact that the algorithms are symmetric with respect to the permutation and negation of any subset of the features of the target concept [Almuallim, 1991].

The results of this experiment for $n = 8, 10$ and 12 are shown in Figure 4.

EXPERIMENT 2: Learning Curve. The purpose of this experiment is to perform a kind of "average-case" comparison between the algorithms. The experiment is conducted as follows. First, we randomly choose a concept such that it has only a few relevant features among many available ones. We then run each of the algorithms on randomly-drawn training samples of this concept and plot the accuracy of the hypothesis (i.e., the percentage of the examples correctly classified by the hypothesis) returned by each algorithm against the training sample size. This is repeated for various sample sizes for the same concept.

The above procedure was applied on 100 randomly selected concepts each having at most 5 relevant features out of 16 available features. For each of these concepts, the sample size m was varied from 20 to 120 examples.

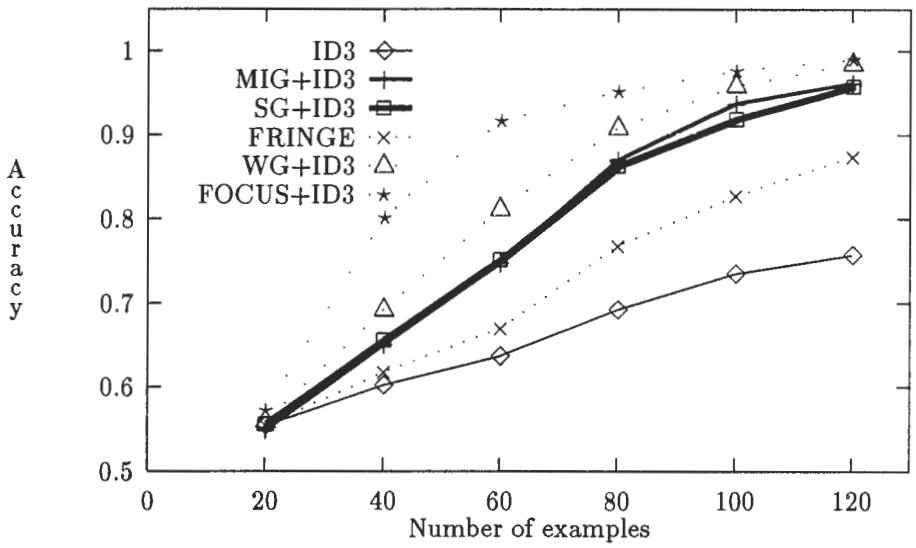


Figure 5: Learning curve for the randomly chosen concept $f(x_1, \ldots, x_{16}) = x_1x_2x_3\bar{x_4} \vee x_1x_2x_3x_4\bar{x_5} \vee x_1x_2\bar{x_3}x_4x_5 \vee x_1\bar{x_2}x_3 \vee x_1\bar{x_2}\bar{x_3}\bar{x_4} \vee \bar{x_1}x_2x_3x_4x_5 \vee \bar{x_1}\bar{x_2}x_3x_4x_5 \vee \bar{x_1}\bar{x_2}x_3\bar{x_4} \vee \bar{x_1}\bar{x_3}x_4x_5 \vee \bar{x_1}\bar{x_3}\bar{x_4}\bar{x_5}$ which has 5 relevant features out of 16.

For each value of m, the accuracy rate was averaged over 100 randomly drawn training samples.

Figure 5 shows a pattern typical of all learning curves that we observed.

As a way to combine the results of the 100 concepts, we have measured the difference in accuracy between FOCUS and each of the other algorithms for each sample size, and averaged that over all the 100 target concepts. The result is shown in Figure 6.

DISCUSSION: The performance of the algorithms tested in the above experiments can be summarized as follows:

1. Each of the three heuristics improved the performance of ID3 when learning in the presence of irrelevant features.

2. Weighted-Greedy gave the overall best approximation to the MIN-FEATURES bias. The performance of this algorithm was quite close to that of FOCUS both in the worst and average cases.

3. Mutual-Information-Greedy and Simple-Greedy are very much alike. These algorithms maintained a reasonable average-case performance, but exhibited a rather bad worst-case performance.

4. Finally, FRINGE showed poor average-case performance, but its worst-case performance is almost as good as Weighted-Greedy and substantially better than Mutual-Information-Greedy and Simple-Greedy. In terms of the computational costs, however, FRINGE is much more expensive than any of the three heuristics considered here.

5.2 Execution Time Comparisons

In this subsection, we compare the computational costs of FOCUS, FOCUS-2 and WG measured as the the number of sufficiency tests and the amount of CPU-time required by each algorithm to return a solution. The three algorithms were implemented in C. Special attention was given to optimizing the implementation of FOCUS.

The experiments were conducted using target concepts that have only a few relevant features out of many available. The relative performance of the algorithms was greatly affected by the problem size measured as the number of the available features and the ratio of the relevant features to that number. However, when the problem size was reasonably large, the relative performance followed a consistent trend. This trend is illustrated by Table 1 where we give the result for a target concept with 9 relevant features out of 25 available features. Training examples were drawn with replacement under the uniform distribution and the training sample size was varied from 100 to 500. The numbers in this table are averaged over 10 runs for each training sample size.

Overall, we found that FOCUS-2 was several times faster than FOCUS and that Weighted-Greedy was further many times faster than FOCUS-2. It is interesting to note that the number of sufficiency tests done by FOCUS remains steady as the training sample size grows, since it blindly follows the same steps for any training sample. FOCUS-2, on the other hand, does a progressively smaller number of sufficiency tests as the number of training examples increases. This is because with a larger sample there is a greater chance of getting conflicts that are explained by only few features, and consequently, a better chance for significant reduction in the number of sufficiency tests needed by FOCUS-2.

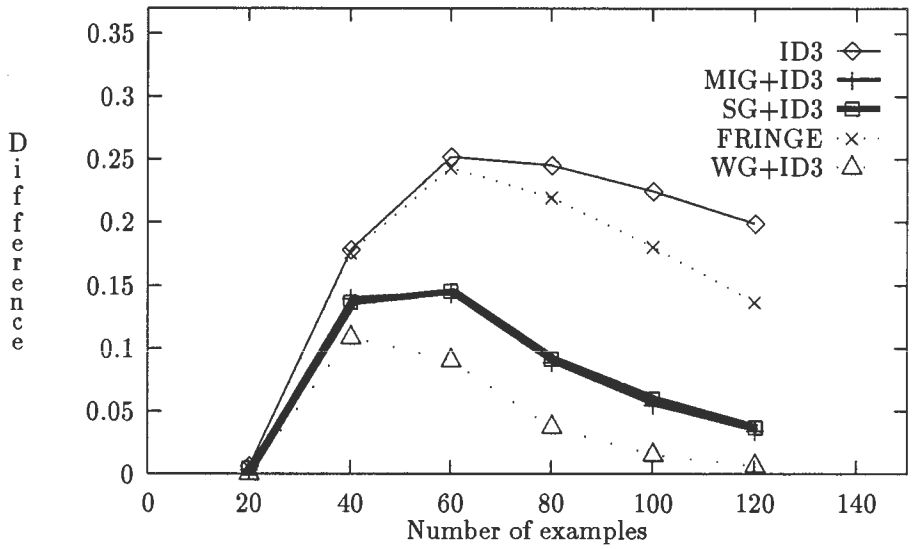


Figure 6: The difference between the accuracy of FOCUS+ID3 and the accuracy of the other algorithms averaged over all the randomly chosen 100 target concepts.

6 Conclusions and Future Research

This paper dealt with the problem of reducing the computational costs involved in implementing the MIN-FEATURES bias. Section 3 introduced the FOCUS-2 algorithm, which provides an implementation of this bias that is substantially faster than the FOCUS algorithm previously given in [Almuallim and Dietterich, 1991]. Section 4 introduced three efficient heuristics for approximating the MIN-FEATURES bias. Experimental studies were reported in which each of these algorithms was used to preprocess the training data to remove the irrelevant features. All of these algorithms were found to be helpful in improving the performance of ID3 in learning tasks where many irrelevant features are present. In particular, the Weighted-Greedy algorithm exhibited excellent performance that closely matches what is obtained by the exact MIN-FEATURES bias. We recommend that, in applications where the MIN-FEATURES bias is appropriate, the Weighted-Greedy algorithm should be applied to preprocess the training sample before invoking a decision-tree algorithm, such as ID3.

All of the approximation algorithms we give can be shown to be polynomial time algorithms. A challenging goal for future research is to prove formal results on the sample complexity of these and similar approximation algorithms.

The work reported in this paper assumes noise-free training data. A direct way to deal with classification noise is to modify the given algorithms by relaxing the requirement of explaining *all* the conflicts generated from the training data. That is, we search for a small set of features that may leave a certain percentage of the conflicts unexplained, where such percentage can be determined through cross-validation. Studying this and more sophisticated approaches to dealing with noise and applying the resulting techniques to real-world problems are two important topics for future work.

References

[Almuallim, 1992] Almuallim, H. Concept Coverage and Its Application to Two Learning Tasks. Ph.D. Thesis. Department of Computer Science, Oregon State University, Corvallis, Oregon. Forthcoming.

[Almuallim and Dietterich, 1991] Almuallim, H. and Dietterich, T. G. Learning With Many Irrelevant Features. In *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91),* 547–552, 1991.

[Almuallim, 1991] Almuallim, H. Exploiting Symmetry Properties in the Evaluation of Inductive Learning Algorithms: An Empirical Domain-Independent Comparative Study. Technical Report, 91-30-09, Dept. of Computer Science, Oregon State University, Corvallis, OR 97331-3202, 1991.

[Blumer *et al.*, 1987] Blumer, A.; Ehrenfeucht, A.; Haussler, D.; and Warmuth, M. Learnability and the Vapnik-Chervonenkis Dimension, Technical Report UCSC-CRL-87-20, Department of Computer and Information Sciences, University of California, Santa Cruz, Nov. 1987. Also in *Journal of ACM,* 36(4):929–965, 1990.

[Ichino and Sklansky, 1984a] Ichino, M. and Sklansky, J. Optimum Feature Selection by Zero-One Integer Programming. In *IEEE Trans. Sys. Man & Cyb.,* Vol. SMC-14, No. 5, 737–746, Sep 1984.

[Ichino and Sklansky, 1984b] Ichino, M. and Sklansky, J. Feature Selection for Linear Classifiers, In *The Seventh International Conference on Pattern Recognition,* 124–127, 1984.

Table 1: The number of sufficiency tests and CPU-time of FOCUS, FOCUS-2 and WG for a target concept with 9 relevant features out of 25 available features.

Algorithm		Training Set Size				
		100	200	300	400	500
FOCUS	Sufficiency Tests	442189.4	1329330.5	2908068.9	2665664.0	2695393.0
	Time (seconds)	9.74	55.1	211.1	191.7	186.1
FOCUS-2	Sufficiency Tests	10593.1	9785.3	11339.5	8768.9	5582.3
	Time (seconds)	2.0	7.1	20.5	31.9	32.5
WG	Sufficiency Tests	8.5	10.1	10.7	10.5	11.0
	Time (seconds)	0.16	0.86	2.31	4.3	7.0

[Garey and Johnson, 1979] Garey, M. R. and Johnson D. S. *Computers and Intractability.* W.H. Freeman and Company, 1979.

[Kittler, 1980] Kittler, J. Computational Problems of Feature Selection Pertaining to Large Data Sets. In *Pattern Recognition in Practice.* Gelsma, E.S. and Kanal, L.N. (eds.) North-Holland Publishing Company, 405–414, 1980.

[Morgera, 1986] Morgera, S.D. Computational Complexity and VLSI Implementation of an Optimal Feature Selection Strategy In *Pattern Recognition In Practice II,* Gelsma, E.S. and Kanal, L.N. (eds.), 1986. Elsevier Science Publishers B.V. (North-Holland), 389–400, 1986.

[Mucciardi and Gose, 1971] Mucciardi, A.N. and Gose, E.E. A Comparison of Seven Techniques for Choosing Subsets of Pattern Recognition Properties, *IEEE Trans. Computers,* Vol. C-20, No. 9, 1023–1031, Sep 1971.

[Narendra and Fukunaga, 1977] Narendra, P.M. and Fukunaga, K. A Branch and Bound Algorithm for Feature Subset Selection, *IEEE Trans. Computers,* Vol. C-26, No. 9, 917–922, 1977.

[Pagallo and Haussler, 1990] Pagallo, G.; and Haussler, D. Boolean feature discovery in empirical learning. *Machine Learning,* 5(1):71–100, 1990.

[Queiros and Gelsma, 1984] Queiros, C.E. and Gelsma, E.S. On Feature Selection, In *The Seventh International Conference on Pattern Recognition,* 128–130, 1984.

[Quinlan, 1986] Quinlan, J. R. Induction of Decision Trees, *Machine Learning,* 1(1):81–106, 1986.

Explicitly Schema-Based Genetic Algorithms

Dwight Deugo and Franz Oppacher
Intelligent Systems Research Group
School of Computer Science, Carleton University
Ottawa, Canada K1S 5B6

Abstract

Genetic Algorithms derive most of their power from the implicit processing of schemata. It is important to note, however, that schemata, as described in the literature, function only as theoretic constructs: they are not processed explicitly. This paper attempts to exploit the power of schemata directly, not just by using them to analyze the performance of a Genetic Algorithm, but by using them as explicit components of the representation. We describe an efficient schema-based representation and operators for solving the Traveling Salesman Problem. In this domain, we found that the direct use of schemata provides a much more natural representation and makes the application of genetic operators easier. For example, by using schemata incomplete tours are permitted, and, because operators don't have to form complete tours, exchange and variation of genetic material is simple. Empirical results show that the new representation and operators quickly provide good results, and we conjecture that explicit schema processing will reduce the complexity of problems in other domains as well.

1 Introduction

Genetic Algorithms (GAs) [Goldberg, 1989; Holland, 1975; Holland *et al.*, 1986; Schaffer, 1989] have demonstrated their robustness and efficiency as search and learning techniques in many application domains. They typically start out with a randomly generated initial population of potential solutions encoded as fixed-length binary strings, i.e. strings over the alphabet {0, 1}. The current population is repeatedly, and in parallel, transformed into a new population by subjecting it to greatly simplified mechanisms of the Darwinian Theory of Evolution by Natural Selection: 1) fitness proportional reproduction, and, 2) genetic operators such as crossover and mutation.

Holland [Holland, 1975] introduced the use of **schemata** (or boolean hyperplanes) to theoretically characterize GAs, and, in particular, to prove the **Schema Theorem**, and to establish that GAs exhibit **'implicit parallelism'**.

Schemata provide a compact way to express similarities among strings, and are themselves strings over the alphabet {*, 0, 1}. The addition of '*' - interpreted as a metalinguistic *don't care* symbol - enables schemata to denote subsets of strings over the alphabet {0, 1} which are similar at certain positions. For example, the schema *101* denotes {01010, 01011, 11011, 11010} whose elements are identical in their second through fourth positions. A schema has several important properties, such as its fitness (as given by the average fitness of the strings it matches), its defining length (the distance between its first and last specific string positions, 2 in the example above), and its order (the number of its fixed positions, 3 in the example above).

The Schema Theorem [Holland, 1975] states that highly fit, short defining length, and low order schemata are allocated exponentially increasing trials in successive generations, under fitness-proportional reproduction, crossover, and mutation. Holland also showed that GAs possess a property of implicit parallelism: while ostensibly processing only the binary strings in the population, they also process in parallel a much greater number of schemata represented in the population. Indeed, for a population of size n at least n^3 schemata are usefully processed without computational overhead.

From these results it is clear that GAs derive most of their power from the implicit processing of schemata. It is important to note, however, that schemata, as described in the literature, function only as **theoretical constructs**, i.e. GAs never actually work with strings containing the '*' symbol, except in classifier systems, but in these systems there is no direct measure of a classifier's fitness, it must be first tested in the environment for its strength to be assigned.

In this paper we show that GAs can indeed benefit from processing schemata explicitly by describing an efficient schema-based representation and operators for solving the **Traveling Salesman Problem** (TSP). Section 2 quickly describes GA research on the TSP. Section 3 introduces our new genetic representation, based on a schema, for the TSP, and discusses how schema fitness is determined. Section 4 describes two genetic schema operators: a mutation operator, and combined inversion-crossover operator. Section 5 discusses our empirical results, and section 6 concludes the paper.

Our approach is further motivated by the often noted fact that a complex structure evolves much faster when it is assembled in hierarchical layers from stable subunits that are

larger than their elementary parts[1]. We therefore conjecture that explicit schema processing will reduce the complexity of problems in other domains as well.

2 GAs and the Traveling Salesman Problem

The Traveling Salesman Problem [Lawler *et al.*, 1985] is a well known NP-hard problem that can only be solved, at best, by heuristic techniques, because methods for finding an exact solution grow exponentially with the number of nodes in the graph. The TSP is of particular interest to the GA research community because of its complex ordering dependencies and its typically non-binary representation. Among many attempts to use traditional GAs on the TSP [Goldberg and Lingle, 1985; Liepins and Hilliard, 1987; Suh and Van Gucht, 1987], only Whitley [Whitley *et al.*, 1989] has met with respectable success.

The lessons learned from GA research on the TSP can be briefly summarized in two statements: 1) use a natural representation that allows easy exchange and variation of genetic material, and, 2) use operators that preserve as much genetic material from the parent strings as possible. The problem with these two statements is that they are in conflict with one another. Any successful attempt at the TSP must come to terms with each of these statements.

When representations such as the path or adjacency representations[2] [Grefenstette *et al.*, 1985] are used, the classical operators of mutation and crossover may no longer generate valid tours. When a representation such as the ordinal representation[3] [Grefenstette *et al.*, 1985], that is closed under crossover and mutation, is used, the resulting tours may have little resemblance to the parent tours. These representational problems are often tackled by introducing modified crossover operators. Although these operators produce valid tours, they create problems of their own. When they swap genetic material between the crossover points, they generate holes - undetermined cities - in the resulting tours. Since they are forced to produce complete tours, they fill the holes, and in the process tend to increase the mutation rate by destroying good edges from their parents. The goal, on the contrary, should be to preserve the maximum amount of information the parents provide [Whitley *et al.*, 1989].

3 Path Schema Representation (PSR)

In order to prevent edge destruction resulting from an operator's attempt to fill in the holes of an offspring, we propose a schema-based representation called the path schema representation (PSR), which tolerates holes.

The PSR is similar to the path representation except that holes (don't care cities, denoted '*') are permitted. For example, (A B * D *) represents a tour from city A to B, to some unknown city, to city D, to some unknown city, and back to city A.

Having accepted the PSR as the representation of a tour, an immediate problem occurs in the calculation of a tour's fitness. The normal method of fitness calculation, for a minimization problem such as the TSP, is as follows:

$$f(x) = C_{max} - g(x) \quad \text{where } g(x) < C_{max}$$
$$= 0 \quad \text{otherwise}$$

C_{max} - the length of the largest possible tour - can be easily computed by multiplying the number of hops in a tour by the greatest distance between two cities plus 1[4], giving a hypothetical worst possible tour length. However, there is a problem in calculating the value of the function $g(x)$ - the length of tour x. Given a tour such as (A B C D E), a tour length can be easily computed[5]. However, how can one compute the length of the tour (A B * D *)? What is the distance between city B and *? Given two similar tours, such as (A B C D E) and (A B * D *), are their fitness values similar? Should they be similar? Should a complete tour's fitness be given more weight? If one is to use the PSR, these questions must be addressed.

After examining three different methods of computing $g(x)$: one that used the maximum distance between any two cities to fill in the distances to unknown cities, and one that used the maximum distances between two cities from the set of unknown cities, we chose to estimate the tour length of a incomplete tour from the present known tour length.

$$g(x) = \frac{\text{(known tour distance * number of hops in tour)}}{\text{(number of known hops in tour)}}$$

For example, the tour (A B C * E) has three known distances: AB, BC, EA, and two unknown distances[6]. If the sum of the known distances is x, the estimated tour length is $\frac{5}{3}x$. If, from an incomplete tour, half of the distances are known, the estimated tour length will be twice of their total. This definition provided a very good estimate of the length of partial tours and provided the exact length of complete tours, however, sometimes partial tour lengths were under estimated, and we wanted to favor complete tours. To solve this problem the following modification was done to $g(x)$:

```
cityBias := 2
IF knownHops <= cityBias THEN
    {Few actual distances between cities are known, use
        worst estimate of tour length}
```

---

1 [Simon, 1969, pp. 90-95], [Simon, 1973].

2 In the path representation, a tour is simply a list of visited cities. In the adjacency representation, a tour consists of a list of cities such that if there is an edge from city *i* to city *j*, then the allele in position *i* is *j*.

3 In the ordinal representation, a tour consists of a list of *N* integers constructed from the path representation in which the *ith* element can range from 1 to *N-i+1*.

4 The one is added to ensure a fitness value >= 1.

5 This is provided that a distance matrix or some formula for computing distances between two cities is given.

6 Trivially, distances CD and DE are the ones that remain, however, assume that it is not known that city D is the one missing.

```
        g(x) := knownDistance +
                (currentWorstDistanceBetweenTwoCities *
                    unknownHops)
    ELSE
        {Mostly know distances between cities, compute
            standard g(x)}
        IF knownHops >= (maxHops - cityBias) THEN
            g(x) := knownDistance *
                    ( maxHops / knownHops )
        ELSE
            {Many unknown distances between cities, penalize
                g(x) }
            g(x) := knownDistance *
                    ( maxHops / (knownHops - cityBias))
```

If very few cities in the tour are known, the worst estimate method is used. If most of the cities in the tour are known, the standard $g(x)$ estimate is used. If the known number of cities does not fall into the previous categories, $g(x)$ is computed as follows:

$$g(x)' = \frac{\text{(known distance * number of hops in tour)}}{\text{(number of known hops in tour - penalty)}}$$

By introducing a penalty (cityBias), this function slightly increases the length of incomplete tours compared to the standard function $g(x)$., thereby, penalizing incomplete tours. We found that a penalty of 2 made partial tours marginally longer than complete tours, but still comparable to them.

Finally, we wanted new, strong tours to be accepted quickly into the population. This was achieved with the following fitness function F(x):

$$F(x) = f(x)^p$$

We found that the best results were achieved with a p value of 2.

4 Operators

The classical TSP operators of mutation and crossover [Grefenstette *et al.*, 1985] will not work with the PSR - or most other TSP representations. For example, simple mutation - the occasional, random change of a gene's allele - cannot be performed because if a gene's allele changes - from one city to another - there will be two genes with the same city, thereby forming an illegal tour. To work properly, operators must contain knowledge of tour ordering and contents dependencies. A simple modification to the mutation operator solves the contents problem: two cities, rather than one, change their locations in the tour. Using this form of mutation, no city is then visited twice. The mutation operator is suppose to add new genetic material, so how does the swapping of cities add any new genetic material to a tour when both cities are still present in the tour? The answer is: by switching cities, new edges are formed - four in total. Edges are the genetic material that is altered during mutation.

Simple crossover suffers from the same fate as simple mutation. The random exchange of genetic material between tours often results, because of the representation, in an invalid tour, as a result of one city occurring more than once in it. To combat the problem, a crossover operator must contain knowledge of tour ordering and contents dependencies if it is to produce legal tours. What genetic material is exchanged during crossover? When the PSR - or most other representations for the TSP - is used, the genetic material exchanged is a subtour. What is a subtour? It is an ordering of edges. Edges and their ordering are the genetic materials that are exchanged during crossover.

One genetic operator that does work with a representation such as PSR is called inversion, and it always produces a legal tour. Inversion selects two cities in the tour and reverses the subtour between them. For example, performing inversion on the tour (A B C D E F) at cities B and E results in the tour (A E D C B F). Since only two cities are actually swapped, it is similar to the modified mutation operator. However, while the modified mutation operator adds four new edges, inversion adds only two new edges - edges AE and BF in the previous example. The inversion operator's purpose is not that of mutation, but rather to produces new tour orderings and as few new edges as possible. Edge ordering is the genetic material maintained during inversion.

We propose two new genetic operators for use with the PSR: the Schema Mutation Operator, and the Inverted Schema Crossover Operator, and make the following goals for them.

1) Operators should focus on the edges, not the cities, for the TSP.
2) Operators should attempt to preserve as many of the parent's original edges in generated offspring.
3) Operators should be considered for edge creation, edge swapping, and edge reordering.
4) Operators should be able function with incomplete tours.

At this point we remind the reader that the PSR is a schema-based representation. Don't care cities are permitted, resulting in the possibility of incomplete tours. Our operators must, therefore, work with both complete and incomplete tours. A complete tour is valid if and only if each city is found once in the tour. An incomplete tour is valid if and only if each city is found once or it is absent from the tour; the don't care city, however, may occur more than once.

4.1 Schema Mutation Operator

Our mutation operator, called the Schema Mutation Operator (SMO), like the classical mutation operator, works on one gene at time. This is made possible by the PSR. Two mutation rates are defined for the operator. The first mutation rate (Pms) represents the rate at which a tour (schema) is considered for mutation. The second mutation rate (Pmg) represents the rate at which a site in the tour is

considered for mutation. Once a tour has been selected for mutation, the operator sweeps the sites one by one and decides which ones to mutate. Once a site is selected for mutation, the operator first collects all existing cities not currently in the tour into a list. Cities may or may not be part of the tour because of the PSR. To this list of cities, the don't care city is added. The new city for the mutation site is then randomly selected from the constructed list. The resulting tour is always valid - only missing cities or the don't care city are added to the tour. If the tour is complete, the first mutation site of the tour will produce only a don't care city - no other cities are currently available for consideration. At the second mutation site of the tour, the don't care city and the city freed by the first mutation are the possible candidate cities for that site. If the tour is initially incomplete, the first mutation my add a 'real' city, because, if the tour is incomplete, there are cities that are not currently in the tour. In summary, any city can mutate to one of the cities not currently in the tour or the don't care city.

There are some immediate benefits of the collaboration between the SMO and the PSR. First, mutation can be done at a single site. Simple mutation for the TSP requires the interaction of at least two or more cities. Secondly, depending on if there are real or don't care cites before and after the current mutation site, and if there is a real or don't care city at the current mutation site, the number of edges changed by one mutation is between zero and two, not four edge changes as in the 2-opt operator [Jog *et al.*, 1989]. Using the SMO, new genetic material is added at a slower, more controlled rate, and all four operator goals are met.

4.2 Inverted Schema Crossover Operator

Our second operator, called the Inverted Schema Crossover Operator (ISCO), provides two operations in one operator: edge swapping, and edge reordering. The crossover rate (Pc) represents the rate at which two selected tours are crossed. After two tours are selected for manipulation, the first portion of the operator constructs an edge list, or graph, of the tours. For example, the edge list of the two tours (A B C D E F) and (B D * A F C) is as follows:

A has edges to:	B F
B has edges to:	A C D
C has edges to:	B D F
D has edges to:	B C E
E has edges to:	D F
F has edges to:	A C E

It is worth mentioning that edges can be duplicated in the tours, e.g. edge AF; some cities can have only two, connecting edges, as a result of the city not being present in one tour, e.g. city E; and some cities have unknown, connecting edges, e.g. edge A *.

After constructing the edge list, the operator then proceeds as follows: while one or more cities exist with more than two edges, it randomly select one of these cities and removes its longest edge from the edge list. This portion of the operator results in a edge list with two interesting properties. First, because it ensures that all cites have two or less edges, only paths, cycles, or disconnected tours are possible - this property is important for the next part of the operator. Secondly, it represents a crossover of the original two tours, were edges have only been remove, not added, and edge removal is biased to remove longer edges. This removal heuristic attempts to shorten the length of the tours created by the operator - an inverse, greedy heuristic.

The final portion of the operator assembles a complete, or partial, tour from the constructed edge list. It begins with an empty tour and proceeds as follows:

```
start with an empty tour
While (one or more cities exist in the edge list with one or
        more edges) do
Begin
    randomly select a city of degree one, or, if there is no
    such city, then select a city of degree 2
    If the city selected has degree 2 Then
        randomly remove one of its edges
    Repeat
        add the city to the new tour;
        follow the other edge to the next city (each city has
            at most degree 1).
        remove edge taken
    Until ( new city has no edge to follow)
End
If complete tour is not constructed Then
    add don't care cities to fill in the tour
```

This operator provides a form of subtour chunking [Grefenstette *et al.*, 1985]. Paths and cycles which represent strong orderings of partial cities are pieced together, randomly, one after another. If either because the original tours were incomplete or the second portion of the operator removed edges that would have completed a tour, a incomplete tour may be constructed. But the operator was made to work with incomplete tours, so this problem is of little concern.

There are some immediate benefits of the ISCO. First, it meets all of our operator goals. Second, the ISCO operates as a knowledgeable crossover operator. By using the heuristic of removing random, long, connecting edges from cities, there is a constant pressure of forming tours of shorter length. Third, by treating the union of edges from the two parent tours as a single set, rather than two separate sets, edge destruction is minimized because few cities are randomly added into the tour. This is in direct contrast with the PMX operator [Oliver et al., 1987], where available cities are used to plug the holes left by the initial crossover procedure. The ISCO also acts as inversion operator. Once an existing subpath or subcycle has been added to a tour, a new one is then added, and so on. Once the initial city of the next subpath is randomly chosen, the cities found in the subpath starting from it are added to the tour. If the chunk is a subpath, there are two possible starting points: city 1 or city n of the subpath. If the chunk is a cycle, there are n cities from which to start the subpath with. The end result is that subpaths can be inverted when added to the tour.

5 Experimental Results

This section describes the results and analysis of using the PSR, SMO and ISCO on four different TSPs: the Karg and Thompson 10 city problem [Karg and Thompson, 1964], the Oliver 30 city problem, and the Eilon 50 and 75 city problems [Whitley *et al.*, 1989]. These problems, described in [Whitley *et al.*, 1989], are part of a package put together by G. Liepins of the Oak Ridge National Laboratory for testing new approaches to the TSP, and we used them in order to make comparisons with existing approaches to the TSP.

For the test runs we used roulette wheel selection; the probability of tour and site mutation was set to 0.1; and the probability of crossover was set to 0.3. The initial population was seeded with tours of order 1; therefore, all destinations in the tour, except one, were initialized to the don't care city. The one, real city in a tour was continually incremented from one tour to the next, resetting to city 1 when the last city was reached. Therefore, at the beginning of the genetic process, the initial population had the maximum number of the smallest building blocks possible, for the given population size. By the end of the genetic process the genetic operators had taken the basic building blocks and constructed larger and larger blocks, finally building complete tours. For each problem, ten test runs were completed for a fixed number of generations. Tables 1 and 2 present the results of the test runs.

Source	N	Best	PSR	Mean
Karg	10	378	378	408
Oliver	30	421	445	486
Eilon	50	428	500	511
Eilon	75	545	622	665

Table 1.

Source	N	Pop-Size	Generations	Edge-Cuts
Karg	10	20	20	86 (7.1%)
Oliver	30	60	60	2781 (8.5%)
Eilon	50	100	100	11531 (7.6%)
Eilon	75	150	150	37689 (7.4%)

Table 2.

Source identifies the problem. *N* indicates the number of cities considered by the problem. *Best* indicates the best known results [Karg and Thompson, 1964; Whitley *et al.*, 1989] of the problem. *PSR* indicates the best tour found in ten test runs using the PSR, SMO and ISCO. *Mean* indicates the mean tour found in the test runs. *Pop-Size* indicates the population sized used for each test run. *Generations* indicates the number of generations permitted for each test run. *Edge-Cuts* identifies the total number of new edges added to the population in the best test run. The addition of new edges is a result of the ISCO finishing one subpath and starting a new one, thereby, creating a new edge The percentage given, estimates the effective mutation rate as a result of the addition of the new edges. It is calculated as follows:

$$\text{effective mutation rate} = \frac{\text{new edges}}{\text{estimated edges manipulated}}$$

$$= \frac{\text{new edges}}{\text{pop-size*generations*Pc*max-edges}}$$

Figures 1 to 4 show the results of the best test run for each of the source problems. In each graph, the average tour length of the population (Average Tour), the actual tour length of the best individual in the population (Current Tour), the estimated tour length of the best individual in the population (Normalized Tour), and the known optimal tour length (Best Tour) are shown at each generation.

Table 3 presents the corresponding results of Whitley. *Whitley* indicates the best tour found in ten test runs using the edge recombination operator [Whitley *et al.*, 1989]. *Crosses* indicates the number of recombinations per run.

Source	N	Whitley	Mean	Pop-Size	Crosses
Oliver	30	421	437	250	3,200
Eilon	50	428	439	600	25,000
Eilon	75	545	559	1,000	80,000

Table 3.

Two factors that can greatly affect the performance of a GA are its population size and the number of generations it is permitted to run. In our tests, we have taken the extreme position of small population sizes and number of generations, which results in a small number of crossovers[7]. This position contrasts with Whitley and others [Jog *et al.*, 1989; Oliver et al., 1987], who use large population sizes and many recombinations for good results. The reason for our position is that we wanted to look at how well our proposed GA does given limited time and computation. The population size and number of generations used in our test runs are the same linear function of the number of cities - two times the number of cities. Test results show very encouraging results. Given these restrictions, the correct tour was found for the Karg and Thompson problem; the tour to the Oliver problem had a relative error of 5.7%; and the tours for the Eilon 50 and 75 city problems had relative errors of 16.8% and 14.1% respectively. Although the optimal tour for each problem was not always found, the graphs indicate that respectable tours are found in approximately 7, 22, 33, and 52 generations, respectively. Subsequent generations improved the tours, but it raises a question: is the extra effort worth the time and computation? As Goldberg points out [Goldberg, 1989], 'the emphasis on convergence is a major flaw in current thinking about search procedures'. Our method produces good, quick results even with the restriction of small population sizes.

Our results also indicate that the ISCO achieves its goal of minimizing the number of new edges it creates, or, inversely, the number of existing edges it destroys. A conservative estimate shows that the effective mutation rate of new edges is below 10%. However, in this case, edge

[7] The number of crossovers is approximately equal to: population size * number of generations * crossover rate.

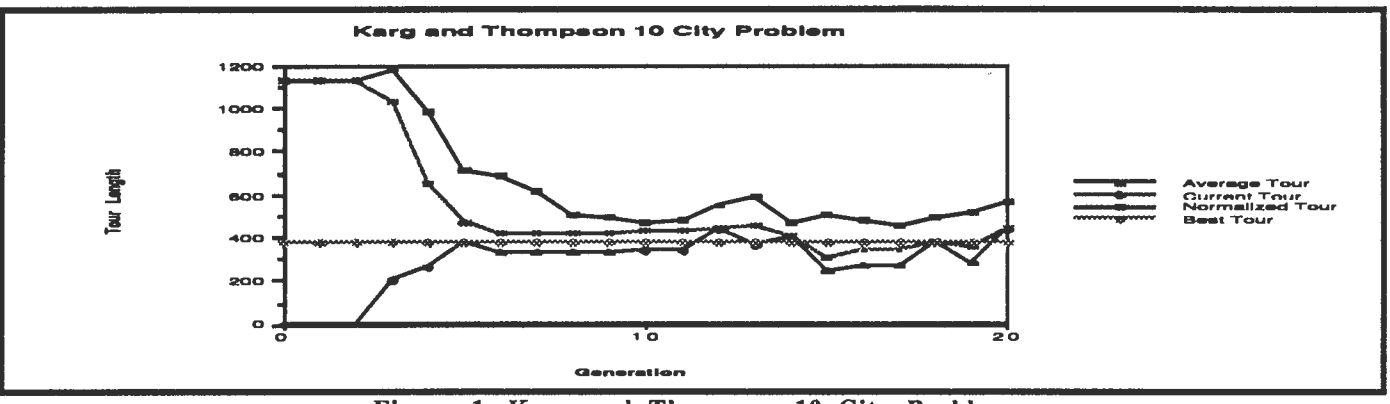


Figure 1. Karg and Thompson 10 City Problem

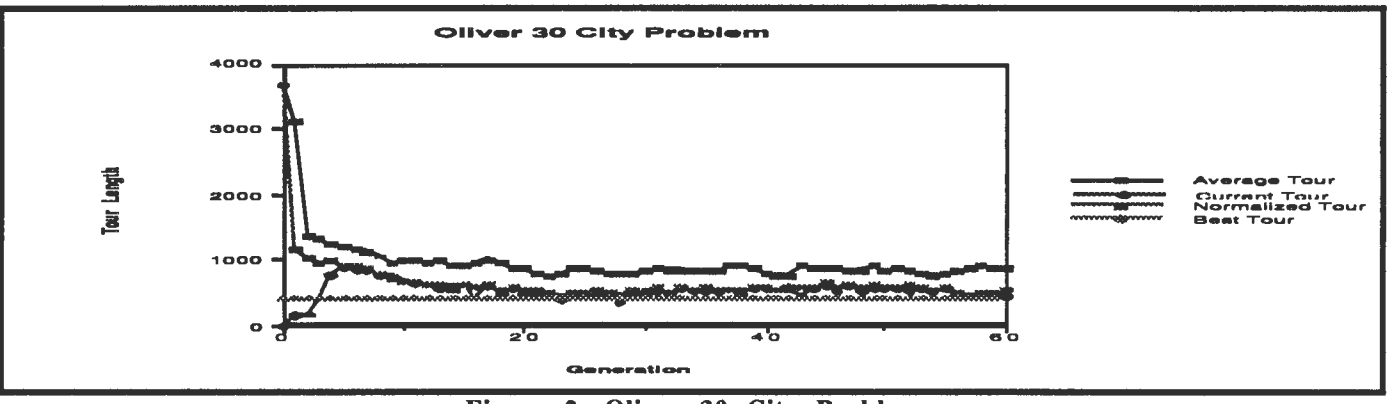


Figure 2. Oliver 30 City Problem

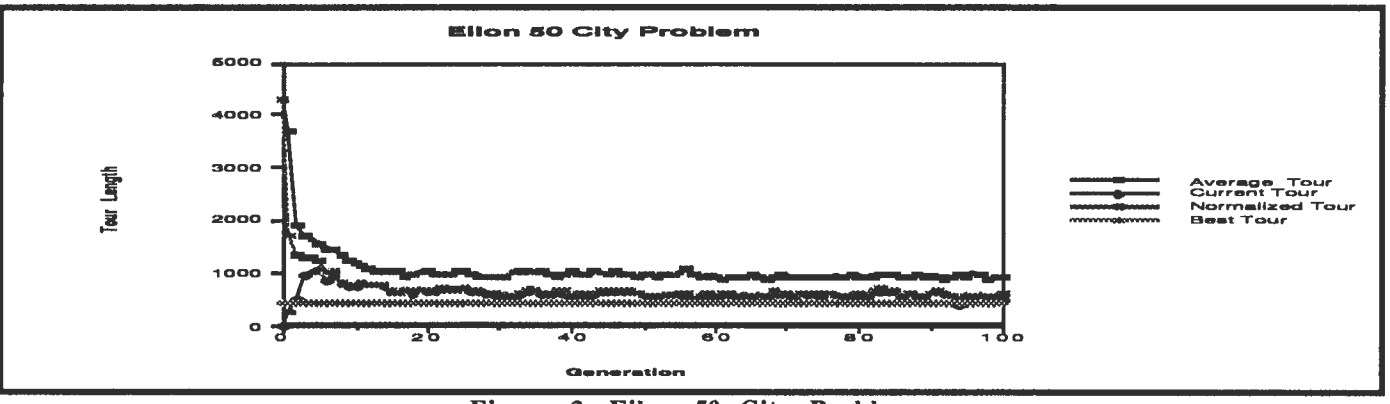


Figure 3. Eilon 50 City Problem

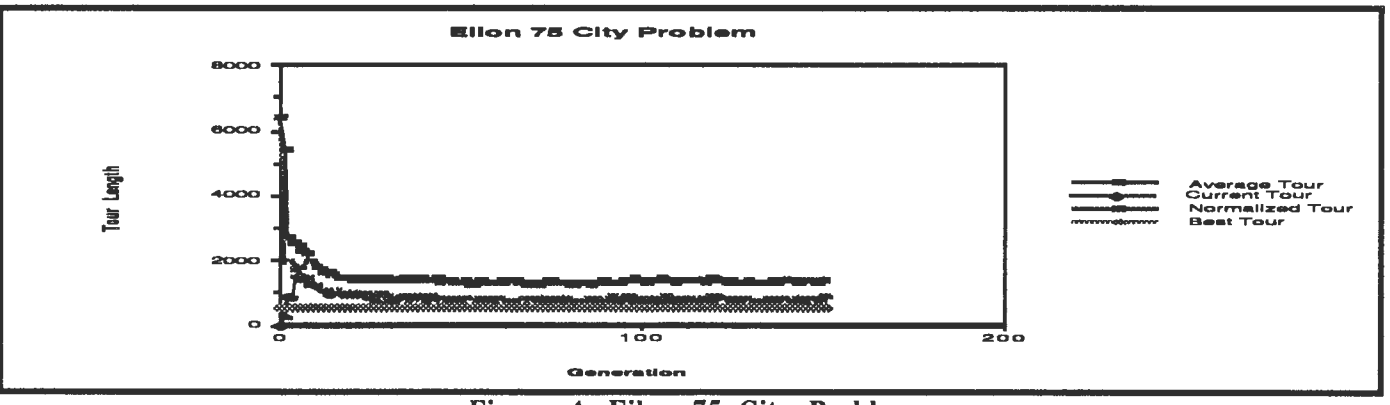


Figure 4. Eilon 75 City Problem

mutations are a result of the ISCO working as an inversion operator, not the SMO. Perhaps, the SMO should only mutate real cities to don't care cities, and rely on the ISCO to perform the edge mutations. The important point is that operators should maintain as many existing edges in tours that they work with, and to not inadvertently increase the edge mutation rate through their operation. We achieve these goals with the PSR and the ISCO.

6 Conclusion

Can GAs benefit from processing schemata explicitly? We believe that they can. There is often a conflict between the use of a natural representation and operators that preserve genetic material. Using a schema-based approach, we have shown that, for the TSP, this conflict can be easily over come. Using the PSR, operators no longer have to form complete tours, but rather just consistent tours. This may seem like a small advantage, but operators that have to form complete tours by randomly filling in the missing pieces of the tour increase the edge mutation rate. If the hypothesis: as many edges as possible should be maintained by an operator, is true, and we believe this is so, the PSR enables us to maintain existing strong subtours, without having them destroyed by the random addition of edges to form weak complete tours.

Another benefit of explicitly working with schemata is that the important genetic material - a schema - is visible to an observer. For example, using the PSR, strong subtours can be easily identified within a tour. If you can see the building blocks, you can work with them. If they are hidden in a complete tour, they are only of immediate use to that tour. No outside agent can ever benefit from that knowledge.

Although more work is required to discover the full potential of working with schemata, the collaboration of the PSR, SMO, and ISCO for the TSP provide us with the incentive, given the good results even with the constraints of time and computation. The ultimate goals of optimization are to seek improved performance to find some optimal point. However, if one can get close to an optimal point using only 20% of the work and time, that point may be sufficient. Our results indicate that we can get close to the optimal point, very quickly, and feel our methods are good for this type of problem, and we conjecture that explicit schema processing will reduce the complexity of problems in other domains as well.

Schemata can be viewed a stable subsystems that are continually combined, forming larger stable subsystems. In our most recent work [Deugo and Oppacher, 1991] we propose to treat strong schemata, at each level of a hierarchy, as stable, nondisruptable units. These units are denoted by new atomic symbols and combined, in turn, into higher-level schemata. Using this view, Simon, then, provides an answer as to why the explicit processing of schemata produces quick results: 'the time required for a complex system to evolve by a process of natural selection is very much shorter if the system is itself comprised of one or more layers of stable components subsystems than if its elementary parts are it only stable components' [Simon, 1973].

References

[Deugo and Oppacher, 1991] D.L. Deugo and F. Oppacher, Computational Evolutionary Epistemology, *Proceedings of the IJCAI-91 Workshop on Evolution and Chaos in Cognitive Processing*, 69-91, 1991.

[Goldberg, 1989] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[Goldberg and Lingle, 1985] D.E. Goldberg and R. Lingle, Alleles, Loci, and the Traveling Salesman Problem, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, Lawrence Erlbaum Associates, 154-159, 1985.

[Grefenstette et al., 1985] J.J. Grefenstette, R. Gopal, and D. Van Gucht, Genetic Algorithms for the Traveling Salesman Problem, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, Lawrence Erlbaum Associates, 160-168, 1985.

[Holland, 1975] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[Holland *et al.*, 1986] J.H. Holland, K.J. Holyoak, R.E. Nisbett, and P.R. Thagard, *Induction: Processes of Inference, Learning, and Discovery*, MIT Press, 1986.

[Jog *et al.*, 1989] P. Jog, J.Y. Suh, and D. Van Gucht, The Effects of Populations Size, Heuristic Crossover and Local Improvement on a Genetic Algorithm for the Traveling Salesman Problem, *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 110-115, 1989.

[Karg and Thompson, 1964] R.L. Karg and G.L. Thompson, A Heuristic Approach To Solving Traveling Salesman Problems, *Management Science*, Vol 10, No 2, 225-248, Jan, 1964.

[Liepins and Hilliard, 1987] G.E. Liepins and M.R. Hilliard, Greedy Genetics, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms and their Applications*, Lawrence Erlbaum Associates, 90-99, 1987.

[Lawler *et al.*, 1985] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan, and D.B. Shmoys, *The Traveling Salesman Problem*, John Wiley & Sons, 1985.

[Oliver et al., 1987] I.M. Oliver, D.J. Smith, and J.R.C. Holland, A Study of Permutation Crossover Operators on the Traveling Salesman Problem, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms and their Applications*, Lawrence Erlbaum Associates, 224-230, 1987.

[Schaffer, 1989] J.D. Schaffer, *Proceedings of the Third International Conference on Genetic Algorithms (ed)*, Morgan Kaufmann, 1989.

[Simon, 1969] H.A. Simon, *The Sciences of the Artificial*, The MIT Press, 1969.

[Simon, 1963] H.A. Simon, The Organization of Complex Systems, *In Hierarchy Theory*, ed. H.H. Pattee, George Braziller, 3-27, 1973.

[Suh and Van Gucht, 1987] J.Y. Suh and D. Van Gucht, Incorporating Heuristic Information into Genetic Search, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms and their Applications*, Lawrence Erlbaum Associates, 100-107, 1987.

[Whitley *et al.*, 1989] G. Whitley, T. Starkweather, D'Ann. Fuquay, Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator, *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 133-140, 1989.

Binary Iterative-Deepening-A*: An Admissible Generalization of IDA* Search

Brian G. Patrick
Department of Computer Science and Engineering
Collège Militaire Royal de St. Jean
Richelain, Québec J0J 1R0
CANADA

Abstract

Iterative-deepening-A* (IDA*) is an admissible heuristic search algorithm which is optimal with respect to space complexity and the cost of solution found over the class of admissible best-first tree search algorithms. However, the optimality of IDA* with respect to time complexity is subject to a number of conditions. In the worst case, IDA* expands $O(N^2)$ nodes where N is the number of nodes that are surely-expanded by A*. To redress this worst case phenomenon of expanding only a few additional nodes over several iterations, a new admissible search algorithm, called Binary IDA* (BIDA*), is presented and compared against the average case performance of IDA* on the Euclidean traveling salesperson problem. It is shown in a small empirical study that BIDA* is a significant improvement over IDA* as both the tour size and the precision of the edge costs increase.

1 Motivation

A wide variety of difficult and often intractable problems in artificial intelligence, operations research and combinatorics are represented in terms of the solution space model and involve a search of that space to find an optimal or near-optimal solution [Ernst and Newell, 1969; Newell and Simon, 1972]. Often, the solution space of a problem is abstractly defined as an implicit and locally-finite graph G with edge costs that are greater than some positive constant δ [Nilsson, 1971; Pearl, 1984]. A single node of G is distinguished as the start node s and represents the initial configuration of a problem. Beginning at the start node s, the selective expansion of nodes explicates the solution space graph G and defines in effect the process of search. Each expansion of a node n generates all immediate successors of n and the costs of the associated edges from n to each of its successors. The search continues until a goal node that satisfies the stated objectives of the problem definition is selected for expansion. Any path from the start node s to a goal node is defined as a solution path and hence, many problems are formulated in terms of the finding the minimum cost or *admissible* path from the start node s to a goal node [Pohl, 1970]

Iterative-deepening-A* (IDA*) is an admissible heuristic search algorithm that combines an optimal utilization of heuristic knowledge with an optimal utilization of memory space [Korf, 1985]. In order to meet these objectives, the IDA* algorithm:

1. Assigns a cost to each node n that is determined by the evaluation function $f(n) = g(n) + h(n)$ where [1]:
   (a) $g(n)$ is the cost of the current path from the start node s to n, and
   (b) $h(n)$ is a non-negative heuristic estimate of $h^*(n)$ where $h^*(n)$ is the cost of the optimal path from n to a goal node. If n is a goal node then $h(n)$ is equal to 0.
2. Performs successive depth-first searches, each rooted at the start node s, that are bounded by increasing values of f.

The cost bound of the initial iteration is equal to the cost of the start node s, that is, $f(s)$. The cost bound of each subsequent iteration is then equal to the cost of the minimum f-value among all generated nodes that exceeded the cost bound of the previous iteration. Given an admissible heuristic function h (i.e. $h \leq h^*$), the following two properties hold for a solution space tree search:

Property 1 *The IDA* algorithm is admissible and the cost bound of the final iteration is equal to the cost of the optimal solution path, henceforth denoted as C^* [Korf, 1988].*

Property 2 *The number of nodes that are expanded by IDA* on its penultimate iteration is equal to the number of nodes that are surely-expanded by A* on G [Patrick, 1991]. By definition, a node n is surely-expanded by A* if and only if there exists a path from the start node s to n along which each node has a cost that is less than C^* [Dechter and Pearl, 1985].*

The latter property establishes a common measure of comparison between the IDA* and A* algorithms. Although the optimality of IDA* with respect to space

[1] The evaluation function $f = g + h$ is historically associated with the A* algorithm [Hart *et al.*, 1968; Hart *et al.*, 1972] where it has been shown that f is an optimal discriminant for additive cost measures [Dechter and Pearl, 1985].

complexity and the cost of solution found are respectively ensured by the nature of depth-first search and the admissibility of h, the IDA* algorithm expands in the worst case $O(N^2)$ nodes where N is the number of nodes that are surely-expanded by A* [Patrick *et al.*, 1992]. The worst case phenomenon arises under the conditions of uniqueness and monotonicity when only a single additional node is selected for expansion from one iteration to the next.

The binary iterative-deepening-A* (BIDA*) algorithm, described in Section 2, is an attempt to redress the worst case conditions of IDA* search without compromising either the admissibility or space optimality of IDA*. In a small empirical study in Section 3, the average case performance of BIDA* is compared against the average case performance of A* and IDA* on the Euclidean traveling salesperson problem (ETSP). In Section 4, BIDA* is contrasted with the IDA*_CR algorithm [Sarkar *et al.*, 1991] which was recently proposed as an alternate approach toward overcoming the worst case time complexity of IDA*. Finally, concluding remarks are offered in Section 5.

2 Description of BIDA*

Binary iterative-deepening-A* (BIDA*) is an admissible generalization of the IDA* algorithm. In light of the worst case scenario of IDA*, the objectives of BIDA* are twofold:

1. To increase the number of additional but admissible nodes that are expanded on each iteration, and
2. To reduce the total number of iterations.

In order to meet these objectives, the cost bound of each iteration of BIDA* is chosen as a point between:

1. A *lower bound* which is non-decreasing from one iteration to the next but remains less than or equal to the cost of the optimal solution path C^*, and
2. An *upper bound* which is non-increasing from one iteration to the next but remains greater than or equal to the cost of the optimal solution path C^*.

Therefore, unlike IDA*, the BIDA* algorithm uses not only a lower bound but also an upper bound in order to establish the cost bound of each successive iteration.

The lower bound of the initial iteration, denoted L_1, is set to the cost of the start node s, that is, $L_1 = f(s)$. The upper bound of the initial iteration, denoted U_1, is set to the cost of any solution path P from the start node s to a goal node q, that is, $U_1 = f(q)$. If the lower bound L_1 is equal the upper bound U_1 then the BIDA* algorithm terminates with the solution path P. Otherwise, the cost bound of the initial iteration is equal to

$$(1-\omega)L_1 + \omega U_1$$

where $0 < \omega < 1$. For the initial iteration and each successive iteration $i \geq 1$, a depth-first search is performed until either one of two conditions is met:

1. A goal node is selected for expansion, or
2. The f-values of all expandable nodes is greater than the cost bound of iteration i, denoted C_i.

If a goal node q is selected for expansion then the upper bound of iteration $i+1$, denoted U_{i+1}, is set to cost of the solution path P from the start node s to q; otherwise, U_{i+1} remains equal to U_i. If, on the other hand, a goal node is not selected for expansion then the lower bound of iteration $i+1$, denoted L_{i+1}, is set to the minimum f-value among all nodes that were generated on iteration i and that exceeded the cost bound C_i; otherwise, L_{i+1} remains equal to L_i. If the upper bound U_{i+1} is equal to the lower bound L_{i+1}, the BIDA* algorithm terminates with the most recent solution path P. If the lower bound remains less than the upper bound then a depth-first search of the solution space is repeated with a cost bound C_{i+1} equal to

$$(1-\omega)L_{i+1} + \omega U_{i+1}$$

where $0 < \omega < 1$.

2.1 Admissibility of BIDA*

Theorem 1 *Given an admissible solution space tree G, BIDA* is admissible.*

Proof: It is sufficient to show that the lower bound of BIDA* will eventually equal but never exceed the cost of the optimal solution path C^* and that the upper bound will eventually equal but never fall below C^*. Therefore, the lower bound can only equal the upper bound at C^* which implies admissibility.

The lower bound of the initial iteration is equal to the cost of the start node s. Since $f(s) \leq f^*(s) = C^*$, the lower bound is less than or equal to C^*. The upper bound of the initial iteration is equal to the cost of any solution path. Since the cost of any solution path is greater than or equal to the cost of the optimal solution path, the upper bound is greater than or equal to C^*. If the initial lower and upper bounds are equal then an optimal solution path is immediately found; otherwise, the cost bound of the initial iteration, denoted C_1, is chosen between the initial lower and upper bounds. If an optimal solution path is not found on iteration $i \geq 1$ then a bounded depth-first search is performed until either a goal node is selected for expansion or the f-values of all expandable nodes is greater than C_i. If a goal node is selected for expansion then the upper bound is set to the cost of the solution path that is found. Clearly, the upper bound remains greater than or equal to C^*. Since the cost bound of the subsequent iteration is less than the new upper bound, each solution path is found at most once. Hence, the cost of each solution path is less than the cost of the previous solution path that is found and the upper bound is less than the previous upper bound. If a goal node is not selected for expansion then the lower bound of the subsequent iteration is set to the minimum f-value among expandable nodes that exceeded the cost bound of iteration i. Hence, the lower bound remains less than or equal to C^* but greater than the previous lower bound. In either case, the interval between the lower and upper bound is reduced from one iteration to the next. Since the cost of each directed edge in G is by definition greater than some positive constant δ, there exists a finite number of nodes whose f-values fall within the interval between the initial lower and upper

bounds. Because each new lower or upper bound is equal to the cost of a node whose f-value falls within the initial interval and because the interval between the lower and upper bound strictly decreases from one iteration to the next, the number of iterations required for either the lower bound or upper bound to reach C^* is finite.

If the lower bound equals C^* before the upper bound then the cost bound of each subsequent iteration remains greater than C^* until an optimal solution path is found. At this point, the upper bound is set to the cost of the optimal solution path and the search terminates. If the upper bound equals C^* before the lower bound then the cost bound of each subsequent iteration remains less than C^* until the lower bound is equal to C^*. At this point, the search terminates and returns the most recent solution path that is found. Since the cost of the most recent solution path is equal to the current upper bound, that is C^*, then an optimal solution path is found.□

2.2 Time Complexity of BIDA*

Lemma 1 *Given an admissible solution space tree G and $0 < \omega < 1$, BIDA* performs at most $\lceil \log_{\frac{1}{1-\varpi}}((U_1 - L_1)10^t + 1) \rceil$ iterations where ϖ is equal to $\min(\omega, 1-\omega)$ and 10^{-t} is equal to the maximum precision of the edge costs.*

Proof: By multiplying the edge costs by 10^t, the lower, upper and cost bounds of each iteration are treated as integral values without loss of information. By definition, the cost bound C_i of iteration i is equal to

$$(1-\omega)L_i + \omega U_i.$$

Since either $L_{i+1} > C_i$ or $U_{i+1} \leq C_i$, the interval between L_i and U_i is reduced by at least a factor of $\varpi = \min(\omega, 1-\omega)$. By reducing the interval from one iteration to the next by at least a factor of ϖ until L_i is equal to U_i, the maximum number of iterations performed by BIDA* given an initial interval $[L_1, U_1]$ is equal to

$$\lceil \log_{\frac{1}{1-\varpi}}((U_1 - L_1)10^t + 1) \rceil.\square$$

Corollary 1 *Given an admissible solution space tree G, the maximum number of iterations performed by BIDA* is minimized at $\omega = 0.5$.*

Corollary 2 *Given an admissible solution space tree G, IDA* performs at most $(C^* - C_1 + 1)10^t$ iterations where C_1 and C^* are the cost bounds of the initial and final iterations, and 10^{-t} is equal to the maximum precision of the edge costs.*

Corollary 3 *If the initial upper bound of an admissible solution space tree is at most a polynomial function of the cost of the optimal solution path then BIDA* performs asymptotically fewer iterations than IDA* in the worst case.*

The lure of fewer iterations, however, does not immediately imply that the time complexity of BIDA* is also less than the time complexity of IDA*. Because the cost bound of each iteration of BIDA* is selected as an arbitrary point between the lower and upper bounds of that iteration, the cost bound is not constrained to be less than or equal to the cost of the optimal solution path. With the potential of expanding several inadmissible nodes on those iterations whose cost bounds are greater than C^*, BIDA* may expand a far greater number of nodes than IDA* notwithstanding the reduction in the number of iterations. However, this computational risk is mitigated in part by one factor: If the cost bound of an iteration is greater than C^* then BIDA* performs a bounded depth-first search *only* until a solution path is found. Therefore, BIDA* does not necessarily perform an exhaustive search of all paths along which each node has an f-value that is less than or equal to the cost bound of the iteration.

3 Empirical Results

Empirical tests that compare the average case performance between the IDA* and BIDA* algorithms are carried out with respect to the Euclidean traveling salesperson problem (ETSP) defined below:

Definition 1 *Given a positive adjacency matrix (c_{ij}) where each element c_{ij} represents the Euclidean distance from city i to city j, the Euclidean traveling salesperson problem (ETSP) is to find the shortest tour that begins at an arbitrary city, visits each other city exactly once and returns to the starting city.*

The ETSP was chosen primarily on the basis of the following observations:

1. In [Patrick *et al.*, 1992], a worst case example of IDA* has been shown on an instance of the asymmetric traveling salesperson problem (ATSP). Similar results noted in [Korf, 1988] also cite the non-optimal performance of IDA* on instances of the TSP.
2. Unlike other common applications such as the 15-Puzzle and the vertex cover problems, the edge costs of the TSP are not necessarily equal to one. Hence, the edge costs may be modeled with arbitrary precision and magnitude.
3. The solution space G_m of an m-city TSP satisfies the following two properties.

   Property 3 *Every terminal node in G_m is located at depth m from the start node s.*

   Property 4 *Every terminal node is a goal node. Therefore, every path in G_m leads to a goal node.*

Properties 3 and 4 ensure that the depth of search is bounded and that a goal node is returned whenever the maximum depth is reached. If the cost bound of an iteration is greater than C^* then BIDA* explores a single path at a time until a solution path P is found. Hence, only those nodes on or before the solution path P are selected for expansion. Since the length of each path is at most m and every path in G_m leads to a goal node, the above properties above help to mitigate the computational risk of potentially expanding several inadmissible nodes.

An instance of an m-city ETSP is generated by randomly selecting m points in the unit square $[0,1]^2$. Each

point $(x_i, y_i) \in [0,1]^2$ represents the position of city i. The Euclidean distance c_{ij} between city i and city j is calculated straightforwardly as

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

for each pair (i, j). Therefore, an instance of the ETSP is characterized by two parameters (m, t) where:

1. m represents the number of cities in the tour, and
2. t represents the maximum precision, evaluated as 10^{-t}, of the Euclidean distances between cities.

For example, the parameters $(9, 5)$ define a 9-city ETSP where 10^{-5} is the maximum precision of the edge costs.

For each m, $5 \leq m \leq 10$, forty random instances of the m-city ETSP are generated. Each instance is solved for a maximum precision of 10^{-t}, $1 \leq t \leq 6$, using the A*, IDA* and BIDA* ($\omega = 0.5$) algorithms. The time complexities of both IDA* and BIDA* are measured in terms of the total number of nodes that are selected for expansion on each iteration leading up to and including the penultimate iteration. Since the time complexity of A* is equal to the number of nodes that are surely-expanded by IDA* on its penultimate iteration and since IDA* is nearly-equivalent[2] to BIDA* for $\omega = 0$, the three algorithms are implemented as a single standard Pascal program. The average time complexity and iteration ratios among A*, IDA* and BIDA* are calculated based on the forty random instances. The results for each parameter (m, t) are recorded in one of the following four tables:

Table 1 (a): The ratio of the average time complexity of IDA* to the average time complexity of A*.

Table 1 (b): The ratio of the average number of iterations performed by BIDA* to the average number of iterations performed by IDA*.

Table 1 (c): The ratio of the average time complexity of BIDA* to the average time complexity of IDA*.

Table 1 (d): The ratio of the average time complexity of BIDA* to the average time complexity of A*.

Each entry in Table 1 represents a ratio of an average performance measure between, say, Algorithm A and Algorithm B. As the tour size and the precision of the edge costs increase, three scenarios are noted:

1. If the ratio remains constant then the performance of A is optimal with respect to the performance of B.
2. If the ratio increases then the performance of A is non-optimal with respect to the performance of B.
3. If the ratio decreases then the performance of B is non-optimal with respect to the performance of A.

In Table 1(a), the non-optimal performance of IDA* on the ETSP is clear. As both the tour size and the precision of the edge costs increase, the ratio of the average

[2]In order to draw an equivalent comparison with IDA*, the lower bound of BIDA* is updated *after* the test for equality with the upper bound.

Size	Maximum Precision, t					
m	1	2	3	4	5	6
5	1.89	3.71	4.06	4.06	4.06	4.06
6	2.20	7.44	9.20	9.48	9.50	9.50
7	2.06	9.66	16.85	18.69	18.88	18.90
8	2.01	12.84	37.90	46.51	47.77	47.92
9	2.02	14.42	69.53	109.18	114.49	115.29
10	1.98	15.12	116.13	607.52	1126.52	1209.80

(a) Average Time Complexities of IDA* to A*

Size	Maximum Precision, t					
m	1	2	3	4	5	6
5	0.827	0.595	0.576	0.580	0.580	0.580
6	0.810	0.424	0.364	0.357	0.357	0.357
7	0.840	0.353	0.273	0.261	0.260	0.261
8	0.841	0.240	0.151	0.137	0.134	0.134
9	0.888	0.217	0.106	0.084	0.082	0.082
10	0.861	0.199	0.075	0.046	0.037	0.035

(b) Average Number of Iterations of BIDA* to IDA*

Size	Maximum Precision, t					
m	1	2	3	4	5	6
5	1.079	0.673	0.652	0.658	0.658	0.658
6	1.022	0.454	0.369	0.358	0.357	0.357
7	1.063	0.359	0.226	0.203	0.201	0.201
8	1.088	0.275	0.131	0.115	0.110	0.109
9	1.162	0.234	0.080	0.050	0.049	0.049
10	0.959	0.218	0.049	0.013	0.008	0.007

(c) Average Time Complexities of BIDA* to IDA*

Size	Maximum Precision, t					
m	1	2	3	4	5	6
5	2.04	2.50	2.65	2.67	2.67	2.67
6	2.25	3.38	3.39	3.40	3.40	3.40
7	2.18	3.47	3.81	3.79	3.79	3.80
8	2.19	3.53	4.98	5.33	5.24	5.24
9	2.34	3.37	5.53	5.45	5.61	5.61
10	1.90	3.29	5.72	8.16	8.84	8.77

(d) Average Time Complexities of BIDA* to A*

Table 1: Average Performance Ratios

time complexity of IDA* to the average time complexity of A* departs quite dramatically from an optimal constant ratio. As expected and as shown in Table 1(b), BIDA* performs on average fewer iterations than IDA* in every instance. Furthermore, the ratio of the average number of iterations performed by BIDA* to those performed by IDA* is decreasing as both the tour size and the precision of the edge costs increase. Hence, the performance of IDA* is non-optimal with respect to BIDA* in terms of the average number of iterations. The reduction in the number of iterations also yields an almost proportional decrease in the average time complexity of BIDA* as shown in Table 1(c). This suggests that the number of inadmissible nodes that are expanded by BIDA* does not significantly impede its performance. Therefore, the average time complexity of IDA* is again non-optimal with respect to the average time complexity of BIDA*. Although the average time complexity ratio between BIDA* and A* continues to increase as both the tour size and the precision of the edge costs increase, the ratio increase in Table 1(d) is comparatively slight. It is therefore encouraging that the reduction in both the number of iterations and the time complexity of BIDA* over IDA* yields a near-optimal performance by BIDA* with respect to A*.

4 Comparison of BIDA* with IDA*_CR

Recently, Sarkar *et al.* developed an admissible version of IDA*, called IDA*_CR, that also redresses the worst case phenomenon of an IDA* search [Sarkar *et al.*, 1991]. The IDA*_CR algorithm differs from BIDA* in two key respects:

1. IDA*_CR performs a depth-first branch and bound search on each iteration as opposed to a strictly depth-first search.
2. The cost bound of each iteration of IDA*_CR is chosen such that the number of additional nodes grows exponentially from one iteration to the next, that is, the heuristic branching factor b_h is constant and greater than one [Korf, 1988].

To guarantee that at least b_h^i additional nodes are expanded between the i and $(i+1)^{st}$ iterations, IDA*_CR uses a set of buckets indexed $1, 2, \ldots, p$ to group the f-values which exceeded the cost bound C_i of the current iteration. Each bucket, denoted B_j, is associated with a mutually-exclusive range of values $[r_j, r_{j+1}]$ where $r_j < r_{j+1}$ for all j, $1 \leq j \leq p-1$. For each node n whose f-value exceeds C_i, the index of bucket B_j is increased by one where

$$r_j < f(n) \leq r_{j+1}.$$

Therefore, the cost bound of iteration $i+1$ is set to the minimum r_{j+1} where the sum of the indices of buckets B_1 through B_j exceeds b_h^i.

It is important to note that an appropriate balance among a) the heuristic branching factor b_h, b) the number of buckets and c) the range of values associated with each bucket must be established. For an inappropriate choice, only the f-values a few generated nodes may fall within the range of values associated with the buckets.

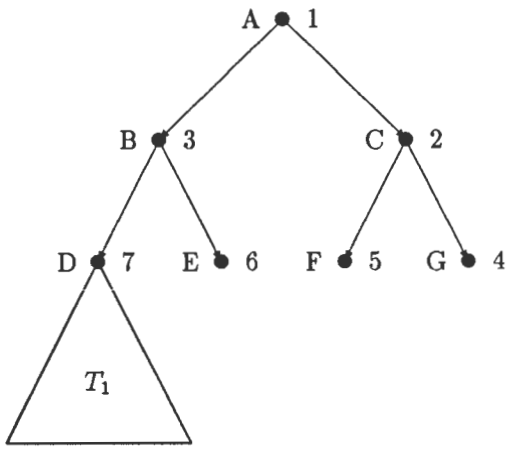


Figure 1: Example of an IDA*_CR Search

In this case, IDA*_CR may not be able to sustain the exponential growth rate from one iteration to the next. Even the choice of an inappropriate b_h alone may lead to a poor performance as shown in Figure 1.

Each node A through G is labeled with its f-value. Nodes A and G are designated as the start and goal nodes respectively. If a heuristic branching factor b_h is chosen as 2 then at least 2^0, 2^1, 2^2, ... additional nodes must be expanded on iterations 1, 2, 3, ... Therefore, the cost bounds of iterations 1, 2, and 3 must equal 1, 3 and 7 respectively. On the third and final iteration, IDA*_CR performs a depth-first branch and bound search until the optimal solution path from node A to node G is found. However, because the cost bound of the third iteration is equal to 7, IDA*_CR may expand an arbitrarily large number of inadmissible nodes in the subtree T_1 rooted at node D.

As shown above, the cost bound of the final iteration of IDA*_CR may exceed the cost of the optimal solution path. Therefore, like the BIDA* algorithm, IDA*_CR potentially expands several inadmissible nodes before an optimal solution path is found. Since both A* and IDA* do not expand any inadmissible nodes for $h \leq h^*$, there is again no common measure of time complexity among A*, IDA* and IDA*_CR that includes the number of nodes that are expanded on the final iteration. Hence, the following claim in [Sarkar *et al.*, 1991, p. 213] is somewhat misleading:

> IDA*_CR expands $O(N)$ nodes where N is the number of nodes that are expanded by A*.

However, because IDA*_CR does not expand any inadmissible nodes on each iteration leading up to and including the penultimate iteration, Theorem 2 may be stated directly.

Theorem 2 *Given an admissible solution space tree G with a constant heuristic branching factor greater than*

*one, IDA*_CR is asymptotically optimal, in terms of the number of nodes that are surely-expanded by A*, over the class of admissible best-first tree search algorithms.*

Unfortunately, a similar claim cannot be made for BIDA*. Nonetheless, it remains to compare how the average time complexity of BIDA* and IDA*_CR are respectively influenced by the expansion of inadmissible nodes.

5 Concluding Remarks

This paper has established two important properties of a BIDA* search on an admissible solution space tree G:

1. BIDA* is admissible.
2. If the initial upper bound of an admissible solution space tree is at most a polynomial function of the cost of the optimal solution path then BIDA* performs asymptotically fewer iterations than IDA* in the worst case.

Unfortunately, the reduction in the number of iterations comes at the expense of potentially expanding a large number of inadmissible nodes. Because, in part, the ETSP satisfies Properties 3 and 4 stated earlier, BIDA* is shown to be a significant improvement over IDA* as both the tour size and the precision of the edge costs increase for the ETSP. However, it remains:

1. To expand the empirical scope to other combinatorial problems and to larger instances of the traveling salesperson problem.
2. To support the empirical work with theoretical justification. For instance, the expected case behaviour of BIDA* may be derived with respect to a probabilistic model of computation that distributes the costs (depths) of the goal nodes over a solution space tree. Such an analysis would help answer an important question: What conditions must the distribution function satisfy in order to ensure the optimal performance of BIDA* with respect to IDA* and better still to A*?
3. To generalize the above analyses for all ω, $0 < \omega < 1$.

References

[Dechter and Pearl, 1985] R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM*, 32(5):505–536, 1985.

[Ernst and Newell, 1969] G.W. Ernst and A. Newell. *GPS: A Case Study in Generality and Problem Solving.* Academic Press, New Year, N.Y., 1969.

[Hart *et al.*, 1968] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. on Systems Science and Cybernetics*, 4:100–107, 1968.

[Hart *et al.*, 1972] P.E. Hart, N.J. Nilsson, and B. Raphael. Correction to: A formal basis for the heuristic determination of minimum cost paths. *SIGART Newsletter*, 37:28–29, 1972.

[Korf, 1985] R.E. Korf. Depth-first iterative-deepening: An optimal admissible search tree search. *Artificial Intelligence*, 27(1):97–109, 1985.

[Korf, 1988] R.E. Korf. Optimal path-finding algorithms. In L. Kanal and V. Kumar, editors, *Search in Artificial Intelligence*, pages 223–267. Springer Verlag, New York, N.Y., 1988.

[Newell and Simon, 1972] A. Newell and H.A. Simon. *Human Problem Solving.* Prentice Hall, Englewood, N.J., 1972.

[Nilsson, 1971] N.J. Nilsson. *Problem Solving Methods in Artificial Intelligence.* McGraw Hill, New York, N.Y., 1971.

[Patrick *et al.*, 1992] B.G. Patrick, M. Almulla, and M.M Newborn. An upper bound on the time complexity of iterative-deepening-A*. *Annals of Mathematics and Artificial Intelligence*, 5, 1992. (to appear).

[Patrick, 1991] B.G. Patrick. *An Analysis of Iterative-Deepening-A*.* PhD thesis, McGill University, Montreal, Quebec, 1991.

[Pearl, 1984] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison Wesley, Menlo Park, CA, 1984.

[Pohl, 1970] I. Pohl. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3):193–204, 1970.

[Sarkar *et al.*, 1991] U.K. Sarkar, P.P. Ghose, and S.C. De Sarkar. Reducing reexpansions in iterative-deepening search by controlling cutoff bounds. *Artificial Intelligence*, 50(2):207–221, 1991.

Probabilistic Hill-Climbing: Theory and Applications

Russell Greiner*
Siemens Corporate Research, Princeton, NJ 08540
greiner@learning.siemens.com (609) 734-3627

Abstract

Many learning systems search through a space of possible performance elements, seeking an element with high expected utility. As the task of finding the globally optimal element is usually intractable, many practical learning systems use hill-climbing to find a local optimum. Unfortunately, even this is difficult, as it depends on the distribution of problems, which is typically unknown. This paper addresses the task of approximating this hill-climbing search when the utility function can only be estimated by sampling. We present an algorithm that returns an element that is, with provably high probability, essentially a local optimum. We then demonstrate the generality of this algorithm by sketching three meaningful applications, that respectively find an element whose efficiency, accuracy or completeness is nearly optimal. These results suggest approaches to solving the utility problem from explanation-based learning, the multiple extension problem from nonmonotonic reasoning and the tractability/completeness tradeoff problem from knowledge representation.

1 Introduction

Many learning tasks can be viewed as a search through a space of possible performance elements seeking an element that is optimal, based on some utility measure. As examples, many inductive systems seek a function whose classification is optimal, *i.e.*, which labels correctly as many examples as possible; and many explanation-based learning [DeJ88, MCK+89] and chunking [LNR87] systems seek a problem solving system that is optimally efficient [Min88, Gre91]. In each of these cases, the utility function used to compare the different elements is defined as the expected value of a particular scoring function, averaged over the distribution of samples (or goals, queries, problems, ...) that will be seen [Hau88, OG90, GO91].

There are two problems with implementing such a learning system: First, we need to know the distribution of samples to determine which element is optimal; unfortunately, this information is usually unknown. There are, of course, standard statistical techniques that use a set of observed samples to estimate the needed information; and several classes of learning systems have incorporated these techniques. For example, many "PAC-learning" systems [Val84] use these estimates to identify an element that is approximately a global optimum.

This leads to the second problem: unfortunately, the task of identifying the globally optimal element, even given the correct distribution information, is intractable for many spaces of elements [Gre91, Hau88]. A common response is to build a system that hill-climbs towards a *local* optimum. Many well-known inductive learning systems, including BACKPROP [Hin89] and ID3 [Qui86], use this approach, as do many speedup learning methods; see especially [GD91]. Unfortunately, few existing systems guarantee that each hill-climbing step is even an improvement, meaning the final element is not always even superior to the initial one, much less an optimum in the space of elements. Moreover, fewer systems include a stopping criterion to determine when the learning has reached a point of diminishing returns.

The work presented here draws ideas from both of these themes: In particular, it describes a general learning algorithm, PALO, that hill-climbs to a local optimum, using a utility metric that is estimated by sampling. Given any parameters $\epsilon, \delta > 0$, PALO efficiently produces an element whose expected utility is, with probability greater than $1 - \delta$, an ϵ-local optimal.[1] Moreover, PALO can work unobtrusively [MMS85], passively gathering the statistics it needs by simply watching a performance element solve problems relevant to a user's applications. Here, the incremental cost of PALO's hill-climbing, over the cost of simply solving performance problems, can be very minor.

*Most of this work was performed at the University of Toronto, where it was supported by the Institute for Robotics and Intelligent Systems and by an operating grant from the National Science and Engineering Research Council of Canada. I also gratefully acknowledge receiving many helpful comments from William Cohen, Dale Schuurmans and the anonymous referees.

[1] Theorem 1 below defines both our sense of efficiency, and "ϵ-local optimality".

Section 2 motivates the use of "expected utility" as a quality metric for comparing performance elements. Section 3 then describes a statistical tool for evaluating whether the result of a proposed modification is better (with respect to this metric) than the original performance element PE; this tool can be viewed as a mathematically rigorous version of [Min88]'s "utility analysis". We use this tool to define the general PALO algorithm, that incrementally produces a series of performance elements $\mathrm{PE}_1, \ldots, \mathrm{PE}_m$ such that each PE_{i+1} is statistically likely to be an incremental improvement over PE_i and, with high confidence, the performance of the final PE_m is a local optimal in the space searched by the learner. Section 4 demonstrates the generality of this approach by presenting three different instantiations of the PALO system, each using its own set of transformations to find a near-optimal element within various sets of performance elements, where optimality is defined in terms of efficiency, accuracy, or completeness, respectively.

2 Framework

We assume as given a (possibly infinite) set of performance elements $\mathcal{PE} = \{\mathrm{PE}_i\}$, where each $\mathrm{PE} \in \mathcal{PE}$ is a system that returns an answer to each given problem (or query or goal, etc.) $q_i \in \mathcal{Q}$, where $\mathcal{Q} = \{q_1, q_2, \ldots\}$ is the set of all possible queries. We also use the utility function $c : \mathcal{PE} \times \mathcal{Q} \mapsto \Re$, where $c(\mathrm{PE}, q)$ measures how well the element PE does at solving the problem q. (Section 4 defines $c_s(\mathrm{PE}, q)$, which quantifies the time PE requires to solve q; $c_a(\mathrm{PE}, q)$, the accuracy of PE's answer; and $c_c(\mathrm{PE}, q)$, PE's categoricity.)

This utility function specifies which PE_i is best for a single problem. Our performance elements, however, will have to solve an entire ensemble of problems. As we obviously prefer the element that is best overall, we must therefore consider the distribution of problems that our performance elements will encounter. We model this using a probability function, $Pr : \mathcal{Q} \mapsto [0, 1]$, where $Pr[q_i]$ denotes the probability that the problem q_i is selected.[2] We then define the *expected utility* of a performance element:

$$C[\mathrm{PE}] \stackrel{def}{=} E[c(\mathrm{PE}, q)] = \sum_{q \in \mathcal{Q}} Pr[q] \times c(\mathrm{PE}, q) \quad (1)$$

Our underlying challenge is to find the performance element whose expected utility is maximal. As mentioned above, there are two problems: First, the problem distribution, needed to determine which element is optimal, is usually unknown. Second, even if we knew that distribution information, the task of identifying the optimal element is often intractable.

3 The PALO Algorithm

This section presents a learning system, PALO (for "Probably Approximately Locally Optimal") that sidesteps the above problems by using a set of sample queries to estimate the distribution, and by hill-climbing efficiently from a given initial PE_0 to one that is, with high probability, essentially a local optimum. This section first states the fundamental theorem that specifies PALO's functionality, then summarizes PALO's code and sketches a proof of the theorem.

[2] We assume that $|\mathcal{Q}|$ is finite for purely pedagogical reasons, as it allows us to define this probability function. There are obvious ways of extending this analysis to handle an infinite set of problems.

Algorithm PALO(PE_0, ϵ, δ)

- $i \leftarrow 0 \quad j \leftarrow 0$

L1: Let $S \leftarrow \{\}$ Neigh $\leftarrow \{ \tau_k(\mathrm{PE}_j) \}_k$

$\Lambda_{max} = \max \{ \Lambda[\mathrm{PE}', \mathrm{PE}_j] \mid \mathrm{PE}' \in \mathrm{Neigh} \}$

L2: Get query q (from the user).

Let $S \leftarrow S \cup \{q\} \quad i \leftarrow i + |\mathrm{Neigh}|$

- If there is some $\mathrm{PE}' \in \mathrm{Neigh}$ such that

$$\Delta[\mathrm{PE}', \mathrm{PE}_j, S] \geq \Lambda[\mathrm{PE}', \mathrm{PE}_j] \sqrt{\frac{|S|}{2} \ln \left(\frac{i^2 \pi^2}{3\delta} \right)} \quad (2)$$

then let $\mathrm{PE}_{j+1} \leftarrow \mathrm{PE}', \quad j \leftarrow j + 1$.

Return to **L1**.

- If $|S| \geq \frac{2\Lambda_{max}^2}{\epsilon^2} \ln \left(\frac{i^2 \pi^2}{3\delta} \right)$ and $\forall \mathrm{PE}' \in \mathrm{Neigh}. \ \Delta[\mathrm{PE}', \mathrm{PE}_j, S] \leq \frac{\epsilon |S|}{2}$, (3)

then halt and return as output PE_j.

- Otherwise, return to **L2**.

Figure 1: Code for PALO

In more detail, PALO takes as arguments an initial PE_0 and parameters $\epsilon, \delta > 0$. It uses a set of sample problems drawn at random from the $Pr[\cdot]$ distribution to climb from the initial PE_0 to a final PE_m, using a particular set of possible transformations $\mathcal{T} = \{\tau_j\}$, where each τ_j maps one performance element to another; see Section 4. PALO then returns this final PE_m. Theorem 1 states our main theoretical results.[3]

Theorem 1 *The* PALO($\mathrm{PE}_0, \epsilon, \delta$) *process incrementally produces a series of performance elements* $\mathrm{PE}_0, \mathrm{PE}_1, \ldots, \mathrm{PE}_m$, *staying at a particular* PE_j *for only a polynomial number of samples before either climbing to* PE_{j+1} *or terminating. With probability at least* $1 - \delta$, PALO *will terminate. It then returns an element* PE_m *whose expected utility* $C[\mathrm{PE}_m]$ *is, with probability at least* $1 - \delta$, *both*

1. *at least as good as the original* PE_0*; i.e.,* $C[\mathrm{PE}_m] \geq C[\mathrm{PE}_0]$*; and*
2. *an* ϵ*-local optimum — i.e.,* $\forall \tau_j \in \mathcal{T}. \ C[\mathrm{PE}_m] \geq C[\tau_j(\mathrm{PE}_m)] - \epsilon$ □.

The basic code for PALO appears in Figure 1. In essence, PALO will climb from PE_j to a new PE_{j+1} if PE_{j+1} is likely to be better than PE_j; *i.e.*, if we are highly confident that $C[\mathrm{PE}_{j+1}] > C[\mathrm{PE}_j]$. To determine this, define

$$d_i = \Delta[\mathrm{PE}_\alpha, \mathrm{PE}_\beta, q_i] \stackrel{def}{=} c(\mathrm{PE}_\alpha, q_i) - c(\mathrm{PE}_\beta, q_i)$$

[3] This proof, and others, appear in the expanded version of this paper [Gre92].

to be the difference in cost between using PE_α to deal with the problem q_i, and using PE_β. As each query q_i is selected randomly according to some fixed distribution, these d_is are independent, identically distributed random variables whose common mean is $\mu = \mathrm{C}[\mathrm{PE}_\alpha] - \mathrm{C}[\mathrm{PE}_\beta]$. (Notice PE_α is better than PE_β if $\mu > 0$.)

Let $Y_n \stackrel{def}{=} \frac{1}{n}\Delta[\mathrm{PE}_\alpha, \mathrm{PE}_\beta, \{q_i\}_{i=1}^n]$ be the sample mean over n samples, where $\Delta[\mathrm{PE}_\alpha, \mathrm{PE}_\beta, S] \stackrel{def}{=} \sum_{q \in S} c(\mathrm{PE}_\alpha, q) - c(\mathrm{PE}_\beta, q)$ for any set of queries S. This average tends to the true population mean μ as $n \rightarrow \infty$; *i.e.*, $\mu = \lim_{n \rightsquigarrow \infty} Y_n$. Chernoff bounds [Che52] describe the probable rate of convergence: the probability that "Y_n is more than $\mu + \gamma$" goes to 0 exponentially fast as n increases; and, for a fixed n, exponentially as γ increases. Formally,[4]

$$Pr[Y_n > \mu + \gamma] \leq e^{-2n\left(\frac{\gamma}{\Lambda}\right)^2}$$
$$Pr[Y_n < \mu - \gamma] \leq e^{-2n\left(\frac{\gamma}{\Lambda}\right)^2}$$

where Λ is the range of possible values of $c(\mathrm{PE}_\alpha, q_i) - c(\mathrm{PE}_\beta, q_i)$. This $\Lambda = \Lambda[\mathrm{PE}_\alpha, \mathrm{PE}_\beta]$ is also used in both the specification of Λ_{max} under Line **L1** and in Equation 2.

The PALO algorithm uses these equations and the values of $\Delta[\mathrm{PE}', \mathrm{PE}_j, S]$ to determine both how confident we should be that $\mathrm{C}[\mathrm{PE}'] > \mathrm{C}[\mathrm{PE}_j]$ (Equation 2) and whether any "$\mathcal{T}$-neighbor" of PE_j (*i.e.*, any $\tau_k(\mathrm{PE}_j)$) is more than ϵ better than PE_j (Equation 3).

We close this section with some general comments on the PALO framework and algorithm.

N-PALO1. The samples that PALO uses may be produced by a user of the performance system, who is simply asking questions relevant to his current applications; here, PALO is unobtrusively gathering statistics as the user is solving his own problems [MMS85]. This means that the total cost of the overall system, that both solves performance problems and "learns" by hill-climbing to successive performance elements, can be only marginally more than the cost of only running the performance element to simply solve the performance problems.

We are using these user-provided samples as our objective is to approximate the average utility values of the elements, over the distribution of problems that the performance element will actually address. This "average case analysis" differs from several other approaches as, for example, we do not assume that this distribution of problems will be uniform [Gol79], nor that it will necessarily correspond to any particular collection of "benchmark challenge problems" [Kel87].

N-PALO2. All three $c_\alpha(\mathrm{PE}, q)$ functions discussed in this paper are "bounded"; *i.e.*, satisfy

$$\forall \mathrm{PE} \in \mathcal{PE}, q \in \mathcal{Q}.\ c_\ell \leq c_\alpha(\mathrm{PE}, q) \leq c_\ell + \lambda$$

for some constants $c_\ell \in \Re$ and $\lambda \in \Re^+$. Here, we can guarantee that $\Lambda[\mathrm{PE}_\alpha, \mathrm{PE}_\beta] \leq \lambda$. For certain transformations τ_k, we can find yet smaller values for $\Lambda[\tau_k(\mathrm{PE}), \mathrm{PE}]$; see [GJ92].

N-PALO3. Although Theorem 1 bounds the number of samples per iteration, it is impossible to bound the number of iterations of the overall PALO algorithm without making additional assumptions about the search space defined by the $\mathcal{T}$ transformations. The theorem's guarantee that PALO will terminate with probability at least $1 - \delta$ requires that the space of performance elements be finite; this is true in all three situations considered in this paper.

N-PALO4. Notice that a "0-local optimum" corresponds exactly to the standard notion of local optimum; hence our "ϵ-local optimum" generalizes local optimality. Notice that PALO's output, PE_m, will (probably) be a real local optimum if the difference in cost between every two distinct performance elements, PE and $\tau(\mathrm{PE})$, is always larger than ϵ. Thus, for sufficiently small values of ϵ, PALO will always produce a *bona fide* local optimum.

N-PALO5. We can view PALO as a variant on *anytime algorithms* [BD88, DB88] as, at any time, PALO provides a usable result (here, the performance element produced at the j^{th} iteration, PE_j), with the property that later systems are (probably) better than earlier ones; *i.e.*, $i > j$ means $\mathrm{C}[\mathrm{PE}_i] > \mathrm{C}[\mathrm{PE}_j]$ with high probability. PALO differs from standard anytime algorithms by terminating on reaching a point of diminishing returns.

Notice finally that PALO will (probably) process more samples using later elements than using the earlier ones, as its tests (Equations 2 and 3) are increasingly more difficult to pass. This behaviour is desirable, as it means that the overall system is dealing with increasing numbers of samples using later, and therefore better, elements.

4 Instantiations of the PALO Algorithm

This section demonstrates the generality of the PALO algorithm by presenting three different instantiations of this framework. For each instantiation, we specify (1) the set of possible performance elements $\mathcal{PE} = \{\mathrm{PE}_i\}$, (2) the set of transformations $\mathcal{T}$ used in the hill-climbing process, and (3) the scoring function $c(\cdot, \cdot)$ used to specify the expected utility. We will also discuss how to obtain the values of $\Lambda[\tau(\mathrm{PE}), \mathrm{PE}]$. (The instantiations of these parameters are also summarized in Table 1.) For pedagogical reasons, each subsection begins with a quick simplistic description of the application, and then provides notes that describe how to build a more comprehensive system.

4.1 Improving Efficiency

Many derivation processes can be viewed as a satisficing search [SK75] through a given graph structure. As an example, notice that using the information shown in Figure 2 to find an answer to the $\mathtt{hop}(\kappa)$ query, for some ground individual κ, corresponds naturally to a search

[4] See [Bol85, p. 12]. *N.b.*, these inequalities holds for essentially *arbitrary distributions*, not just normal distributions, subject only to the minor constraint that the sequence $\{\Delta_i\}$ has a finite second moment.

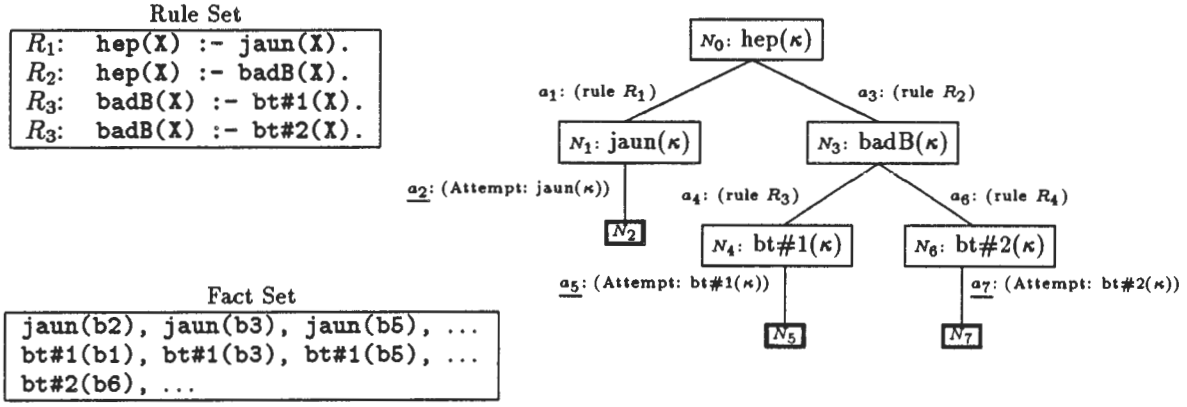


Figure 2: "Inference Graph" G_A, used by Θ_0 and Θ_1

through the G_A inference graph (formed from a given set of rules) seeking a successful database retrieval.[5] A *strategy* specifies the order in which to perform the various rule-based reductions (*e.g.*, the a_1 arc reduces the N_0: `hep(`κ`)` goal to the N_1: `jaun(`κ`)` subgoal, based on the rule R_1) and the database retrievals (*e.g.*, the a_2 arc from N_1 to N_2 corresponds to the attempted database retrieval `jaun(`κ`)`). We can express each strategy as a sequence of G_A's arcs; *e.g.*, the strategy

$$\Theta_0 = \langle a_1, a_2, a_3, a_4, a_5, a_6, a_7 \rangle$$

corresponds to the obvious depth-first left-to-right traversal, with the understanding that the performance element using this strategy will stop whenever it reaches a "success node" (*e.g.*, if the a_2 retrieval succeeds, then Θ_0 reaches the success node N_2 and so stops with success), or has exhausted all of its reductions. (Figure 2 doubly-boxes G_A's success nodes, N_2, N_5 and N_7.) There are many other possible strategies, including

$$\Theta_1 = \langle a_3, a_4, a_5, a_6, a_7, a_1, a_2 \rangle,$$

as well as non–depth-first strategies, etc.

Each strategy will find an answer, if one exists. As this is a satisficing search, all answers are equally acceptable [SK75], which means that all strategies are equally accurate. We can therefore consider the costs of the strategies, preferring the one whose expected cost is minimal.

Letting $f_i \in \Re^+$ be the positive cost of traversing the a_i, we can compute the $c_s(\Theta, q)$, the cost of using strategy Θ to find an answer to the query q. For example, $c_s(\Theta_0, \texttt{hep(b2)}) = f_1 + f_2$, $c_s(\Theta_0, \texttt{hep(b1)}) = f_1 + f_2 + f_3 + f_4 + f_5$, and $c_s(\Theta_1, \texttt{hep(b1)}) = f_3 + f_4 + f_5$. (These different strategies have different costs for a given query as each stops as soon as it finds an answer.) The *expected cost*, of course, depends on the distribution of queries; *i.e.*, on how often the query will be `hep(b1)`, versus `hep(b2)`, etc. Moreover, the task of finding the *globally* optimal strategy is NP-hard [Gre91].

[5] Here, `hep(`χ`)` means χ has hepatitis, `jaun(`χ`)` means χ is jaundiced, and `badB(`χ`)` means χ has "bad blood", `bt#i(`χ`)` means χ tests positive for blood test #i.

This looks like a job for PALO.[6] We first define the set of reordering transformations $\mathcal{T}^{RO} = \{\tau_{\alpha 1,\alpha 2}\}$, where each $\tau_{\alpha 1,\alpha 2}$ maps one strategy to another by moving the subgraph under the $\alpha 1$ arc to before $\alpha 2$ and its subgraph. For example, $\tau_{a_3,a_1}(\Theta_0) = \Theta_1$, and $\tau_{a_6,a_4}(\Theta_0) = \langle a_1, a_2, a_3, \boxed{a_6, a_7}, a_4, a_5 \rangle$. PALO also needs to compute $\Lambda[\tau(\Theta_\alpha), \Theta_\alpha]$; these values are bounded by $c(G) = \sum_i f_i$, the sum of the costs of all of the arcs in the inference graph G; see Note N-Eff2 below.

N-Eff1. This class of performance elements corresponds to many standard problem solvers, including PROLOG [CM81]; see also [GN87]. We can also use these inference graphs to describe operators working in state spaces; here each internal arc of the inference graph corresponds to an operator invocation and each leaf arc to a general "probabilistic experiment". Using G_A, for example, a_3 could encode the "take some blood" operator, and a_5, the experiment that succeeds if the patient tests positive on `bt#1`, etc.

N-Eff2. The companion paper [GJ92] provides more formal descriptions of inference graphs and strategies. That article also presents an efficient analytic way of computing upper and lower bounds of $\Delta[\tau_{\alpha 1,\alpha 2}(\Theta_j), \Theta_j, S]$ (which can be used in Equation 2 and 3, respectively), based only on running Θ_j; this provides a way of obtaining good estimates of $\Delta[\tau_{a1,a2}(\Theta_j), \Theta_j, S]$ *without* first constructing and then executing each $\tau_{a1,a2}(\Theta_j)$ over all $S = \{q_i\}$ queries. It also presents empirical evidence that a system that uses those estimates can still work effectively.

That paper also discusses how this instantiation of the PALO algorithm fits into the framework of "explanation-based learning" systems, and in particular, argues that it provides a mathematical basis for [Min88]'s "utility analysis".

[6] Of course, all of the signs in Figure 1 should be flipped, as we are here measuring *cost* rather than utility, and so prefer the element with *minimal*, rather than maximal, cost. Note also that we are viewing each strategy as a performance element.

4.2 Improving Accuracy

A nonmonotonic system can be ambiguous, in that it can produce many individually plausible but collectively incompatible solutions to certain queries [Rei87]. Unfortunately, only (at most) one of these solutions is correct; the challenge then is to determine which one. This is the essence of the "multiple extension problem" in knowledge representation [Rei87, HM86, Mor87], and corresponds to the "bias problem" in machine learning [Mit80, Utg84, RG87, Hau88]. This subsection addresses this problem by seeking a credulous system, related to the given initial nonmonotonic system, that is "optimally correct"; *i.e.*, which produces the correct answer most often.

In more detail, we assume there is a correct answer to each query q, denoted $\mathcal{O}[q]$; hence $\mathcal{O}[2+2 = \texttt{?x}] = \texttt{Yes}[\texttt{?x} \rightsquigarrow \texttt{4}]$. Each correct answer is either "Yes" (possibly with a binding list, as shown here) or "No". Using $\text{PE}(q)$ to represent the answer returned by the credulous performance element PE, we can define the utility function

$$c_a(\text{PE}, q) \stackrel{def}{=} \begin{cases} +1 & \text{if } \text{PE}(q) = \mathcal{O}[q] \\ 0 & \text{if } \text{PE}(q) = \texttt{IDK} \\ -1 & \text{otherwise} \end{cases} \quad (4)$$

where IDK represents "I don't know".

We focus on stratified THEORIST-style performance elements [PGA86] [Prz87, Bre89, vA90], where each element $\text{PE} = \langle \mathcal{F}, \mathcal{H}, \Theta \rangle$ corresponds to a set of factual information $\mathcal{F}$, a set of allowed hypotheses $\mathcal{H}$ (each a simple type of default [Rei87]) and a specific ordering of the hypotheses. As a specific example, consider $\text{PE}_A = \langle \mathcal{F}_0, \mathcal{H}_0, \Upsilon_A \rangle$, where[7]

$$\mathcal{F}_0 = \left\{ \begin{array}{l} \forall x.\ \texttt{E}(x)\ \&\ \texttt{N}_E(x) \Rightarrow \texttt{S}(x, \texttt{G}) \\ \forall x.\ \texttt{A}(x)\ \&\ \texttt{N}_A(x) \Rightarrow \texttt{S}(x, \texttt{W}) \\ \forall x.\ \neg\texttt{S}(x, \texttt{G}) \vee \neg\texttt{S}(x, \texttt{W}) \\ \texttt{A}(\texttt{Z}),\quad \texttt{E}(\texttt{Z}),\quad \ldots \end{array} \right\} \quad (5)$$

is the fact set;

$$\mathcal{H}_0 = \left\{ \begin{array}{ll} h_1: & \texttt{N}_E(x) \\ h_2: & \texttt{N}_A(x) \end{array} \right\}$$

is the hypothesis set, and $\Upsilon_A = \langle h_1, h_2 \rangle$ is the hypothesis ordering.

To explain how PE_A would process a query, imagine we want to know the color of Zelda — *i.e.*, we want to find a binding for ?c such that $\sigma =$ "S(Z, ?c)" holds. PE_A would first try to prove σ from the factual information $\mathcal{F}_0$ alone. This would fail, as we do not know if Zelda is a normal elephant or if she is a normal albino (*i.e.*, whether $\texttt{N}_E(\texttt{Z})$ or $\texttt{N}_A(\texttt{Z})$ holds, respectively). PE_A then considers using some hypothesis — *i.e.*, it may assert an instantiation of some element of $\mathcal{H}_0$ if that proposition is both consistent with the known facts $\mathcal{F}_0$ and if it allows us to reach a conclusion to the query posed. Here, PE_A could consider asserting either $\texttt{N}_E(Z)$ (meaning that Zelda is a "normal" elephant and hence is colored *Gray*) or $\texttt{N}_A(Z)$ (meaning that Zelda is a "normal" albino and hence is colored *White*). Notice that either of these options, individually, is consistent with everything we know, as encoded by $\mathcal{F}_0$. Unfortunately, we cannot assume both options, as the resulting theory $\mathcal{F}_0 \cup \{\texttt{N}_E(Z), \texttt{N}_A(Z)\}$ is inconsistent.

We must, therefore, decide amongst these options. PE_A's hypothesis ordering Υ_A specifies the priority of the hypotheses. Here $\Upsilon_A = \langle h_1, h_2 \rangle$ means that h_1 : $\texttt{N}_E(x)$ takes priority over h_2 : $\texttt{N}_A(x)$, which means that PE_A will return the conclusion associated with $\texttt{N}_E(Z)$ — *i.e.*, Gray, encoded by $\texttt{Yes}[\texttt{?c} \mapsto G]$, as $\mathcal{F}_0 \cup \{\texttt{N}_E(Z)\} \models \texttt{S(Z, G)}$.[8]

Now consider the $\text{PE}_B = \langle \mathcal{F}_0, \mathcal{H}_0, \Upsilon_B \rangle$ element, which differs from PE_A only in terms of its ordering: As PE_B's $\Upsilon_B = \langle h_2, h_1 \rangle$ considers the hypotheses in the opposite order, it will return the answer $\texttt{Yes}[\texttt{?c} \mapsto \texttt{W}]$ to this query; *i.e.*, it would claim that Zelda is white.

Which of these two elements is better? If we are only concerned with this single Zelda query, then the better (read "more accurate") PE_i is the one with the larger value for $c_a(\text{PE}_i, \texttt{S(Z, ?c)})$; *i.e.*, the PE_i for which $\text{PE}_i(\texttt{S(Z, ?c)}) = \mathcal{O}[\texttt{S(Z, ?c)}]$. In general, however, we will have to consider a less trivial distribution of queries. To illustrate this, imagine Equation 5's "..." corresponds to $\{\texttt{A}(\texttt{Z}_1), \texttt{E}(\texttt{Z}_1), \ldots, \texttt{A}(\texttt{Z}_{100}), \texttt{E}(\texttt{Z}_{100})\}$, stating that each $\texttt{Z}_i$ is an albino elephant; and that the queries are of the form "$\texttt{S}(\texttt{Z}_i, \texttt{?c})$", for various $\texttt{Z}_i$s.

The best PE_i now depends on the distribution of queries (*i.e.*, how often each "$\texttt{S}(\texttt{Z}_i, \texttt{?c})$" query is posed) and also on the correct answers (*i.e.*, for which $\texttt{Z}_i$s $\mathcal{O}[\texttt{S}(\texttt{Z}_i, \texttt{?c})] = \texttt{Yes}[\texttt{?c} \mapsto \texttt{W}]$ as opposed to $\mathcal{O}[\texttt{S}(\texttt{Z}_i, \texttt{?c})] = \texttt{Yes}[\texttt{?c} \mapsto \texttt{G}]$, or some other answer). That is, it depends on the expected accuracy of each system $\text{C}_a[\text{PE}_i]$, which is defined by plugging Equation 4's $c_a(\cdot, \cdot)$ function into Equation 1. We would then select the PE_i system with the larger $\text{C}_a[\cdot]$ value.

In general, $\text{PE} = \langle \mathcal{F}, \mathcal{H}, \Upsilon \rangle$ can include a much larger set of hypotheses $\mathcal{H} = \{h_1, \ldots h_n\}$. As before, each ordering $\Upsilon = \langle h_{\pi(1)}, \ldots h_{\pi(n)} \rangle$ is a sequence of $\mathcal{H}$'s elements. PE's uses this information when answering queries: Let i be the smallest index such that $\mathcal{F} \cup \{h_i\}$ is consistent and $\mathcal{F} \cup \{h_i\} \models q/\lambda_i$ for some answer λ_i; here PE returns this λ_i. If there are no such i's, then PE returns IDK.

Our goal is identifying the ordering that is accurate most often. Unfortunately, the task of identifying this optimal ordering of the hypotheses is NP-complete even for the simplistic situation we have been considering (where every derivation involves exactly one hypothesis, etc.); see [Gre92].

Once again, PALO is designed to deal with this situation. We first define the set of transformations $\mathcal{T}^A = \{\tau_{ij}\}_{i,j}$, where each τ_{ij} moves the j^{th} term in the ordering to just before the i^{th} term — *i.e.*, given any ordering $\Upsilon = \langle h_1, h_2, \ldots, h_n \rangle$,

[7] Here Z refers to Zelda, $\texttt{A}(\chi)$ means χ is an albino, $\texttt{E}(\chi)$ means χ is an elephant, and $\texttt{S}(\chi, \phi)$ means χ's color is ϕ. The first three clauses in Equation 5 state that normal elephants are gray, normal albinos are white, and (in effect) that S is a function.

[8] This uses the instantiation $\texttt{S(Z, G)} = \texttt{S(Z, ?c)/Yes}[\texttt{?c} \mapsto G]$. We will also view "$q$/No" as "$\neg q$".

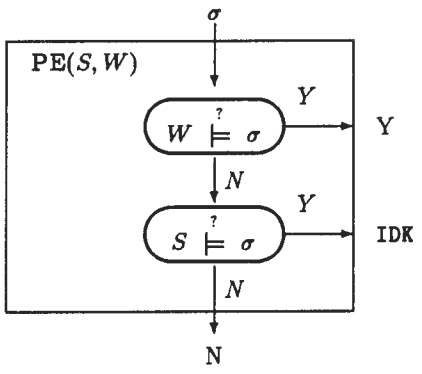


Figure 3: Flow Diagram of $\mathrm{PE}(S, W)$ addressing $\Sigma \stackrel{?}{\models} \sigma$

$\tau_{ij}(\Upsilon) = \langle h_1, \ldots, h_{i-1}, h_j, h_i, \ldots, h_{j-1}, h_{j+1}, \ldots, h_n \rangle$. We can compute the value of $\Delta[\tau_{ij}(\Upsilon_k), \Upsilon_k, S]$ for each τ_{ij} transformation and each set of queries S based on whether $\mathcal{F} \cup \{h_\ell\} \stackrel{?}{\models} q/\mathcal{O}[q]$ for each hypothesis h_ℓ. (Hence, each step of the overall PALO computation is efficient if this test is polynomial time, *e.g.*, if we are dealing with propositional Horn theories, etc.) Observe finally that $\Lambda[\mathrm{PE}_\alpha, \mathrm{PE}_\beta] \leq 2$ for all $\mathrm{PE}_\alpha, \mathrm{PE}_\beta$.

N-Acc1. In many situations, we may want to consider each hypothesis to be the conjunction of a *set* of sub-hypotheses, which must all collectively be asserted to reach a conclusion. Here, we can view $\mathcal{H} = \mathcal{P}[H]$ as the power set of some set of "sub-hypotheses", H.

N-Acc2. Our descriptions have assumed that every ordering of hypotheses is meaningful. In some contexts, there may already be a meaningful partial ordering of the hypotheses, perhaps based on specificity or some other criteria [Gro91]. Here, we can still use PALO to complete the partial ordering, by determining the relative priorities of the initially incomparable elements.

N-Acc3. The motivation underlying this work is similar to the research of [Sha89] and others, who also use probabilistic information to order the various default rules. Our work differs by providing a way of obtaining the relevant statistics, rather than assume that they are known *a priori*.

4.3 Improving Categoricity

The task of determining whether a query is entailed by a theory is known to be intractable if the theory is a general propositional theory. It can, however, be performed efficiently if the theory contains only Horn clauses. Selman and Kautz [SK91] use this observation to define a particular "knowledge compilation" method: Given a general propositional theory Σ, their compiler computes a pair of "bracketing" Horn theories S and W, with the property $S \models \Sigma \models W$.[9] The resulting "compiled system" PE = $\mathrm{PE}(S, W)$ uses these bracketing theories to determine whether a query σ follows from Σ, as shown in Figure 3: If $W \models \sigma$, PE terminates with "yes"; otherwise, if $S \not\models \sigma$, then PE terminates with "no". (Notice that these are the correct answers, in that $W \models \sigma$ guarantees that $\Sigma \models \sigma$, and $S \not\models \sigma$ guarantees that $\Sigma \not\models \sigma$. Moreover, these tests are linear in the sizes of σ and S (respectively, σ and W) [DG84].) Otherwise, if $W \not\models \sigma$ and $S \models \sigma$, PE returns IDK. Notice this compiled system is usually tractable,[10] yet can deal with an arbitrary propositional theory. It may, however, no longer be completely categoric; hence, we have (potentially) sacrificed complete accuracy for tractability [SK91].

We of course would like to find an approximation $\langle S, W \rangle$ that minimizes the probability that the associated $\mathrm{PE}(S, W)$ system will return IDK. To state this more precisely: Given any approximation $\langle S, W \rangle$ and query σ, let $c_c(\langle S, W \rangle, \sigma) \stackrel{def}{=} d(W, \sigma) + (1 - d(S, \sigma))$ where

$$d(S, \sigma) \stackrel{def}{=} \begin{cases} 1 & \text{if } S \models \sigma \\ 0 & \text{otherwise .} \end{cases}$$

Hence, $c_c(\langle S, W \rangle, \sigma) = 1$ if σ is "covered" by $\langle S, W \rangle$, in that either $W \models \sigma$ or $S \not\models \sigma$. Using Equation 1, we can then define $\mathrm{C_c}[\langle S, W \rangle]$ to be the expected value of $c_c(\langle S, W \rangle, \cdot)$. Our goal is to determine the approximation $\langle S, W \rangle$ with the largest $\mathrm{C_c}[\cdot]$ value. As before, this task is NP-hard (see [Gre92]) and depends on the distribution, suggesting yet again that we use the PALO system.

Observe that the set of queries covered by a strengthening and a weakening are disjoint — *i.e.*, for any approximation $\langle S, W \rangle$, there is no query σ such that both $W \models \sigma$ and $S \not\models \sigma$. This means an approximation $\langle S_i, W_j \rangle$ is, with probability at least $1 - \delta$, within ϵ of a local optimum if S_i (resp., W_j) is within $\epsilon/2$ of a locally optimal strengthening (resp., weakening) with probability at least $1 - \delta/2$. We can therefore decouple the task of finding a good strengthening from that of finding a good weakening, and handle each separately. This paper discusses only how to finding a good strengthening; [Gre92] merges this with the algorithm that computes a good weakening.

We are seeking a strengthening S_{opt} whose $\mathrm{D}[S_{opt}]$ value is minimal, where $\mathrm{D}[S_{opt}] = E[d(S, \cdot)]$ is the expected value of $d(S, \cdot)$. (Recall we want $S_{opt} \models \sigma$ to *fail* for as many queries as possible.) It is easy to see that this S_{opt} should be a weakest strengthening; *i.e.*, satisfy $\mathrm{OptS}(\Sigma, S_{opt})$ where

$$\mathrm{OptS}(\Sigma, S) \iff S \models \Sigma \;\&\; \mathrm{Horn}(S) \;\&\; [\neg\exists T.\ [S \models T \models \Sigma \;\&\; \mathrm{Horn}(S) \;\&\; S \not\equiv T]$$

To compute these OptSs: Define a "horn-strengthening" of the clause $\gamma = \{a_1, \ldots, a_k, \neg b_1, \ldots, \neg b_\ell\}$ to be any maximal clause that is a subset of γ and is Horn — *i.e.*, each horn-strengthening is formed by simply discarding all but one of γ's positive literals. Here, there are k horn-strengthenings of γ, each of the form $\gamma_j = \{a_j, \neg b_1, \ldots, \neg b_\ell\}$.

[9] We call each such S a "Strengthening" of the initial theory, and each such W an "Weakening". This subsection deal with clausal theories; each such theory is a set (conjunction) of clauses, where each clause is a set (disjunction) of atomic literals, each either positive or negative. A theory is Horn if each clause includes at most one positive literal.

[10] Note N-Cat2 below explains this caveat.

	Efficiency	Accuracy	Categoricity
Performance Elements $\mathcal{PE}$	satisficing strategies	hypothesis orderings	Horn-strengthenings
Utility function $c(\cdot,\cdot)$	computation time	$\mathrm{PE}(q) \stackrel{?}{=} \mathcal{O}[q]$	$S \stackrel{?}{\models} q$
Tranformations $\mathcal{T}$	reorder arcs	reorder priority	change 1 clause
Range $\Lambda[\tau(\mathrm{PE}), \mathrm{PE}]$	$\leq c(G)$	≤ 2	≤ 1

Table 1: Summary of Applications

Now write $\Sigma = \Sigma_H \cup \Sigma_N$, where Σ_H is the subset of Σ that are Horn and $\Sigma_N = \{\gamma^i\}_{i=1}^m$ is its non-Horn subset. [SK91] proves that each optimal strengthening is of the form $S_o = \Sigma_H \cup \Sigma'_N$, where each $\gamma' \in \Sigma'_N$ is a horn-strengthening of some $\gamma \in \Sigma_N$. By identifying each Horn-strengthened theory with the "index" of the positive literal used (*i.e.*, $\gamma^i{}_j = \{a^i_j, \neg b^i_1, \ldots, \neg b^i_{\ell(i)}\}$), we can consider any Horn-strengthened theory to be a set of the form $S_{\langle j(1), j(2), \ldots, j(m)\rangle} = \Sigma_H \cup \{\gamma^1_{j(1)}, \gamma^2_{j(2)}, \ldots, \gamma^m_{j(m)}\}$.

We can navigate about this space of Horn-strengthened theories by incrementing or decrementing the index of a specific non-Horn clause: That is, define the set of $2m$ transformations $\mathcal{T}^{ID} = \{\tau^+_k, \tau^-_k\}_{k=1}^m$ where each τ^+_k (resp., τ^-_k) is a function that maps one strengthening to another, by incrementing (resp., decrementing) the "index" of k^{th} clause — *e.g.*, $\tau^+_k(S_{\langle 3,9,\ldots,i_k,\ldots,5\rangle}) = S_{\langle 3,9,\ldots,i_k+1,\ldots,5\rangle}$, and $\tau^-_i(S_{\langle 3,9,\ldots,i_k,\ldots,5\rangle}) = S_{\langle 3,9,\ldots,i_k-1,\ldots,5\rangle}$. (Of course, the addition and subtraction operations wrap around.)

This instantiation of the PALO process starts with any given Horn-strengthened theory (perhaps $S_{\langle 1,1,\ldots,1\rangle}$) and hill-climbs in the space of Horn-strengthened theories, using this set of $\mathcal{T}^{ID}$ transformations. As $\Delta[\tau^\pm_k(S_i), S_i, \sigma]$ depends only on whether $\tau^\pm_k(S_i) \models \sigma$ and $S_i \models \sigma$, it can be answered efficiently, as S_i and all $\tau^\pm_k(S_i)$ are Horn. (In fact, this process can also use the support of σ from S_i to further improve its efficiency.) Notice finally that $\Lambda[\tau^\pm_k(S_i), S_i] \leq 1$ for all strengthenings S_i and all $\tau^\pm_k \in \mathcal{T}^{ID}$.

N-Cat1. The $\mathrm{PE}(S_i, W_j)$ systems discussed here each return IDK if $W \not\models \sigma$ and $S \models \sigma$. [Gre92] proposes several other options for this situation — *e.g.*, perhaps the PE should "guess" at an answer here, or perhaps spend as long as necessary to compute whether $\Sigma \stackrel{?}{\models} \sigma$, etc. — and discusses their relative advantages.

N-Cat2. [Gre92] also presents an algorithm that finds a good weakening. For subtle reasons, that process is slightly different from PALO, and computes a W_g that is close to the *global* optimal, with high probability.

(Unfortunately, the size of the optimal weakening can be exponential in the size of the initial theory, meaning the linear bounds mentioned above are not meaningful. [Gre92] considers ways of finding weakenings that are good with respect to a utility metric that combines both categoricity and efficiency [GE91], to produce a polynomially-sized weakening.)

5 Conclusion

This paper first poses two of the problems that can arise in learning systems that seek a performance element whose expected utility is optimal [Hau90, Vap82]: *viz.*, that the distribution information (which is required to determine which element is optimal) is usually unknown, and that finding a globally optimal performance element can be intractable. It then presents the PALO algorithm that side-steps these shortcomings by using statistical techniques to approximate the distribution, and by hill-climbing to produce a near locally optimal element. After defining this algorithm and specifying its behaviour, we demonstrate its generality by showing how it can be used to find a near-optimal element in three very different settings, based on different spaces of performance elements and different criteria for optimality: efficiency, accuracy and categoricity. (See Table 1.) These results suggest approaches to solving the utility problem from explanation-based learning, the multiple extension problem from nonmonotonic reasoning and the tractability/completeness tradeoff problem from knowledge representation.

References

[BD88] M. Boddy and T. Dean. Solving time dependent planning problems. Technical report, Brown University, 1988.

[Bol85] B. Bollobás. *Random Graphs.* Academic Press, 1985.

[Bre89] G. Brewka. Preferred subtheories: An extended logical framework for default reasoning. In *Proceedings of IJCAI-89*, Detroit, 1989.

[Che52] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sums of observations. *Annals of Mathematical Statistics*, 23:493–507, 1952.

[CM81] W. Clocksin and C. Mellish. *Programming in Prolog.* Springer-Verlag, New York, 1981.

[DB88] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of AAAI-88*, 1988.

[DeJ88] G. DeJong. AAAI workshop on Explanation-Based Learning. Sponsored by AAAI, 1988.

[DG84] W. Dowling and J. Gallier. Linear time algorithms for testing the satisfiability of propositional horn formula. *Journal of Logic Programming*, 3:267–84, 1984.

[GD91] J. Gratch and G. Dejong. A hybrid approach to guaranteed effective control strategies. In *Proceedings of IWML-91*, 1991.

[GE91] R. Greiner and C. Elkan. Measuring and improving the effectiveness of representations. In *Proceedings of IJCAI-91*, 1991.

[GJ92] R. Greiner and I. Jurišica. a statistical approach to solving the EBL utility problem. In *Proceedings of AAAI-92*, 1992.

[GN87] M. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann Publishers, Inc., 1987.

[GO91] R. Greiner and P. Orponen. Probably approximately optimal derivation strategies. In *Proceedings of KR-91*, 1991.

[Gol79] A. Goldberg. An average case complexity analysis of the satisfiability problem. In *Proceedings of CADE-79*, 1979.

[Gre91] R. Greiner. Finding the optimal derivation strategy in a redundant knowledge base. *Artificial Intelligence*, 50(1):95–116, 1991.

[Gre92] R. Greiner. Probabilistic hill-climbing: Theory and applications. Technical report, Siemens Corporate Research, 1992.

[Gro91] B. Grosof. Generalizing prioritization. In *Proceeding of KR-91*, April 1991.

[Hau88] D. Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 1988.

[Hau90] D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. Technical Report UCSC-CRL-91-02, UC Santa Cruz, 1990.

[Hin89] G. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40(1-3):185–234, 1989.

[HM86] S. Hanks and D. McDermott. Default reasoning, nonmonotonic logics, and the frame problem. In *Proceedings of AAAI-86*, 1986.

[Kel87] R. Keller. Defining operationality for explanation-based learning. In *Proceedings of AAAI-87*, 1987.

[LNR87] J. Laird, A. Newell, and P. Rosenbloom. SOAR: An architecture of general intelligence. *Artificial Intelligence*, 33(3), 1987.

[MCK+89] S. Minton, J. Carbonell, C. Knoblock, D. Kuokka, O. Etzioni, and Y. Gil. Explanation-based learning: A problem solving perspective. *Artificial Intelligence*, 40(1-3):63–119, 1989.

[Min88] S. Minton. *Learning Search Control Knowledge: An Explanation-Based Approach.* Kluwer Academic Publishers, 1988.

[Mit80] T. Mitchell. The need for bias in learning generalizations. Technical Report CBM-TR-117, Laboratory for Computer Science Research, May 1980.

[MMS85] T. Mitchell, S. Mahadevan, and L. Steinberg. LEAP: A learning apprentice for VLSI design. In *Proceedings of IJCAI-85*, 1985.

[Mor87] P. Morris. Curing anomalous extensions. In *Proceedings of AAAI-87*, 1987.

[OG90] P. Orponen and R. Greiner. On the sample complexity of finding good search strategies. In *Proceedings of COLT-90*, 1990.

[PGA86] D. Poole, R. Goebel, and R. Aleliunas. Theorist: A logical reasoning system for default and diagnosis. Technical Report CS-86-06, University of Waterloo, 1986.

[Prz87] T. Przymusiński. On the declarative semantics of stratified deductive databases and logic programs. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216, 1987.

[Qui86] J. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[Rei87] R. Reiter. Nonmonotonic reasoning. In *Annual Review of Computing Sciences*, volume 2, Annual Reviews Incorporated, 1987.

[RG87] S. Russell and B. Grosof. A declarative approach to bias in concept learning. In *Proceedings of AAAI-87*, 1987.

[Sha89] L. Shastri. Default reasoning in semantic networks: A formalization of recognition and inheritance. *Artificial Intelligence*, 39:283–355, 1989.

[SK75] H. Simon and J. Kadane. Optimal problem-solving search: All-or-none solutions. *Artificial Intelligence*, 6:235–247, 1975.

[SK91] B. Selman and H. Kautz. Knowledge compilation using horn approximations. In *Proceedings of AAAI-91*, 1991.

[Utg84] P. Utgoff. *Shift of Bias for Inductive Concept Learning.* PhD thesis, Rutgers, Laboratory for Computer Science Research, October 1984.

[vA90] P. van Arragon. Nested default reasoning with priority levels. In *Proceedings of CSCSI-90*, 1990.

[Val84] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–42, 1984.

[Vap82] V. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, New York, 1982.

Synthesis of System Configurations Based on Desired Behavior

Michel Benaroch
MIS Group, School of Management
Syracuse University
Syracuse, NY 13244

Vasant Dhar
Information Systems Department
New York University
New York, NY 10006

Abstract

In design, inferring structure from function is a combinatorial generate-and-test problem. Existing methods use pre-stored domain-specific partial configurations to constrain the generator. We have found that for certain types of economic and physical systems consisting of two-terminal components connected in parallel, it is fruitful to specify function in terms of desired behavior, and to identify sets of structurally connected components whose combined behavior under specific operating conditions matches that desired behavior. In this paper, we present a qualitative synthesis technique which constructs all system configurations with a given desired behavior. It uses two synthesis operators called *stretch* and *steepen* that operate on qualitatively specified piecewise linear functions that characterize the behavior of components. Our technique is domain independent. We are currently applying it in the domain of financial hedging, where behaviors of the components (stocks, bonds, options, etc.) are specified in terms of two-dimensional piecewise linear relationships, and the goal is to synthesize these to produce a constrained behavior in response to uncontrollable economic events. We are also investigating the use of our technique in physical domains through the configuration of analog computers.

1 Introduction

Many design problems can be formulated as a process of search in which design parameters are constrained to produce system configurations with some desired functionality [Mittal and Araya, 1986]. When the search space of alternative configurations is immense, it is more reasonable to construct configurations rather than prestore them for selection. In such cases, the design problem entails synthesis of configurations.

Synthesis of configurations involves searching the space of permutations of elementary components in a domain. This search problem is combinatorial, and can be shown to be NP-complete. To solve this problem one must therefore use a search process which makes use of good heuristics.

This paper presents a qualitative synthesis technique that we have developed to solve a synthesis problem that is similar to the one above. This problem involves only systems that are configured from two-terminal components (i.e., one input and one output nodes) connected in parallel, where each component has associated with it a number of two-dimensional piecewise linear relationships that characterize its behavior in specified regions. Such relationships are used commonly in economics to model financial instruments, such as bonds, options, etc. More importantly, these relationships are used as a basis for evaluating and managing the risk associated with uncontrollable factors such as interest rates, currency exchange, and so on. Specifically, if the problem goal and constraints are specified in terms of two-dimensional piecewise linear relationships, the search problem is one of permuting these piecewise linear functions in order to satisfy the constraints. Conceptually, the problem could be formulated as a linear programming problem although such a formulation would be difficult to solve. For this reason, heuristics based on the structure of constraints are important for achieving good solutions.

Our qualitative synthesis technique searches a space of two-dimensional piecewise linear functions, each modeling the behavior of one elementary component under its operational regions. The technique constructs configurations by permuting such piecewise linear functions and comparing them against another piecewise linear function that specifies the desired behavior of a system we seek to synthesize.

Our technique uses two means to constrain the search process. One is a qualitative abstraction over the piecewise linear functions modeling the behavior of components with similar functionality. This reduces significantly the number of piecewise linear functions to be permuted. The other means is knowledge about algebraic operations on two-dimensional piecewise linear functions that are used to create permutations of such functions, and about stretching and/or steepening such a function over one of its definitional regions. This knowledge is used to eliminate many permutations associated with configurations that do not satisfy the problem goal.

The paper is organized as follows. Section 2 reviews the complexity of configuration problems, and introduces the reader to the domain of *financial hedging* which involves a configuration problem similar to the one discussed here. Section 3 explains our solution approach to the problem of configuration based on desired behavior. Section 4 extends Kuipers' [1986] notion of qualitative behavior to define the *qualitative functional behavior*, the *ql-function*, and the *qualitative configuration* of a two-terminal system. Section 5 presents our

synthesis algorithm and defines heuristic operators on ql-functions, which constrain the search process used by our algorithm. Section 6 discusses limitations of our technique and directions for future work.

2 Background

2.1 Analysis & Design of Physical Systems

Much of the work on qualitative reasoning about physical systems is based on modeling the relationships between: *structure* — a collection of components connected as a system; *behavior* — a sequence of states of a system and its components over some time-interval; and *function* — the purpose of structure in producing the behavior of a system. These relationships can be summarized as follows. The behavior of a system results from interactions between the behavior of its components. The effects of a change in the state of a component propagates locally through structural connections causing a change in the state of other components and of the system as a whole. The function of a system, on the other hand, explains in terms of causality why and how structure of a system determines its behavior [De Kleer and Brown, 1984]. From now on, the paper uses the term configuration to refer to the structure of a system.

The goal of analysis techniques (e.g., qualitative simulation) is to infer behavior from structure. Given a structural description of a system and its initial state, these techniques predict the qualitative transitions that a system makes over time. Some techniques describe the transitions of a system from one state to another using a number of two-dimensional piecewise linear plots (e.g., [Kuipers, 1986]).

The primary goal of design techniques is to infer the structure of a system from its function. Inferring structure from function is a generate-and-test search problem. In order to constrain the generator, many design techniques use pre-stored knowledge about configurations of systems that are specific to one domain. For example, to generate configurations in the domain of paper transportation, Mittal and Araya [1986] use hierarchical decompositions of top-level functional goals (e.g., "design a driver role") into subgoals that are each associated with pre-stored partial configurations. However, in the lack of such contextual knowledge, inferring structure from function is an overwhelmingly complex task.

For some design situations it makes sense to specify function in terms of the desired behavior of a system over some operational regions, that is, in terms of input values that a system can accept and corresponding output values it should produce. For example, the desired behavior of a hydraulic servo valve is *"If the applied electrical signal is null the valve should be closed. Otherwise, the valve opening should be proportional to the applied electrical signal."* Note that this kind of behavior can be expressed as a two-dimensional piecewise linear function.

This suggests that in some situations it may be possible to eliminate the need to pre-store design configurations by looking at the problem of inferring structure from behavior. Given the desired behavior of a system, the goal is to identify elementary components in the domain which can be connected in such a way that the overall behavior resulting from interactions between their behaviors according to causality laws is identical to the desired behavior. However, since the number of configurations (i.e., permutations of components) is infinite, it is necessary to construct configurations by means of synthesis, rather than select them by means of classification.

2.2 Hedge Design

In the domain of *financial hedging* functionality is described qualitatively in terms of desired behavior, and the structure and behavior of domain components — instruments (e.g., stocks, bonds, options) — is modeled very much like that of physical systems [Elton and Gruber, 1987; Hart *et al.*, 1986]. Hedging is concerned with the design of vehicles that can protect against losses, or generate profits, under future uncertain events. The primary design goal of hedging is to configure all hedge vehicles that provide a certain *payoff-profile* [Benaroch and Dhar, 1991]. A *payoff-profile* specifies qualitatively what a trader is willing to pay and risk based on his predictions of how the behavior of a certain economic factor is likely to change over some period, and his assessments of how this change will effect the behavior (i.e., market price) of instruments. To derive a payoff from such assessments, a trader looks for hedge vehicles that provide a payoff-profile that takes into account his predictions.

For example, if a trader believes that over the next three months the price of stock S will increase above h_1 but not above h_2, s/he can define the "ratio-spread" payoff-profile in Figure 1. This payoff-profile states that in case of a price increase of S above h_1, but not above h_2, the trader would like to make a profit, and in case of a price increase above h_2, the trader is willing to take a loss. One vehicle that provides that payoff-profile involves buying a call option and selling two call option with exercise prices h_1 and h_2, respectively.[1] Buying a call with exercise price h_1 ensures that the trader will not lose money if the price of S moves below h_1, and ensures that the trader will make money by buying stocks for h_1 when stock price is actually between h_1 and h_2. As long as the price of S is not above h_2, the trader profits from the sale of two calls to another party which believes otherwise.[2]

We consider a *hedge vehicle* to be a two-terminal system whose behavior is described by a qualitative piecewise linear function (i.e., payoff-profile).[3] A *generic vehicle* involves selling or buying instruments of one type. A *compounded vehicle* involves selling and/or buying instruments of more than one type. In effect, a com-

[1] The buyer of a *call option* on a stock with exercise price s has the right, but not the obligation, to buy from the call seller that stock at the exercise price s at some future expiration date.

[2] The amount of profit/loss is determined by the exact calls the trader buys/sells and their quantities. These are selected based on the trader's risk tolerance and the degree of belief in her predictions.

[3] Although a hedge vehicle is actually a multi-terminal system, hedging traders are only interested in the behavior of its value as a function of the parameter being hedged (e.g., interest rate, stock price). Since this behavior changes across market situations, we use qualitative simulation [Kuipers, 1986] to derive it in each market situation by perturbing only the parameter being hedged.

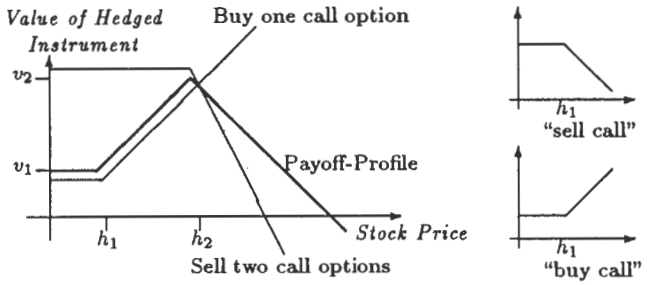


Figure 1: A "ratio-spread" payoff-profile

pounded vehicle is a linear combination of two or more generic vehicles, and its payoff-profile is a linear combination of payoff-profiles provided by generic vehicles. At any moment, a trader can construct thousands of generic vehicles, each with a different payoff-profile, and virtually an infinite number of compounded vehicles, some of which have the same payoff-profile.

The problem facing traders in the configuration phase is similar to the one faced in many other design situations. Given the payoff-profile (behavior) of every generic vehicle (elementary component), synthesize the configuration of all compounded vehicles whose payoff-profile matches some goal payoff-profile.

3 Solution Approach

One can construct system configurations by applying algebraic operations on the transfer functions of elementary domain components [Saucedo and Schiring, 1968]. Given the transfer functions of two components, their algebraic sum produces the transfer function of a system made from the two components connected in parallel, and their product produces the function of a system made from the two components connected in series.

Since the input to the configuration problem discussed here is in terms of desired behavior, the above concept can be applied on the two-dimensional piecewise linear functions describing the behavior of two components over their operational regions. Given that the desired behavior of a prospective system, and the actual behavior of every elementary component over its operational regions is expressed as a two-dimensional piecewise linear function (hereafter, pl-function), system configurations can be synthesized by searching the space of all permutations of elementary pl-functions. The goal of this search is to find all linear combinations of elementary pl-functions that produce the goal pl-function. Although this problem can be formulated using linear programming, it would be difficult to formulate and solve it when the number of elementary pl-functions is large (50 or more).

As the goal behavior is described qualitatively, one may suggest that the number of elementary pl-functions can be reduced by replacing all pl-functions with similar shape by one qualitative pl-function (hereafter, ql-function). This is equivalent to the grouping of all components with similar functionality into one class. One possible representation of a ql-function for a two-terminal system that is characterized by parameters p_1 and p_2 is: $\text{sign}(\frac{dp_2}{dp_1}) = \begin{cases} qdir_1 & p_1 \in qval_1 \\ \dots & \dots \\ qdir_n & p_1 \in qval_n \end{cases}$, where $qdir \in \{-1, 0, 1\}$ is the qualitative direction-of-change of p_2, and $qval$ is a value bounding a qualitative range on the real-line. However, by describing behavior qualitatively the specifity of behavior of each elementary component of the same class is lost. Thus, the use of linear programming is no longer feasible. It will not be able, for example, to find configurations involving combinations of two or more different pl-functions that are represented by the same ql-function.

We have developed a technique that solves this synthesis problem symbolically. Our technique uses a qualitative abstraction over the behavior of components with similar functionality. It also uses two heuristic synthesis operators which enables rediscovering much of the information lost by the abstraction of similar detailed behaviors into one qualitative behavior. These operators are used to stretch and steepen a ql-function over its definitional regions. In Figure 1, for example, the goal ql-function is synthesized by stretching the ql-function for a "sell call" over the region $(0, h_2)$, and steepening it over the region (h_2, ∞).

4 Qualitative Behavior & Configuration

This section uses the notion of a qualitative behavior [Kuipers, 1986] to define the a *qualitative functional behavior*, a *ql-function*, and a *qualitative configuration* of a two-terminal physical system. The reader is referred to Kuipers [1986] for a discussion of Definitions 4.1 – 4.5.

4.1 Qualitative Behavior

A physical system is characterized by multiple real-valued continuously time-varying parameters. A parameter is a *reasonable function*, $f : [a, b] \rightarrow \Re^*$, $\Re^* = [-\infty, \infty]$ is the extended real-line[4]. A reasonable function, f, has a finite totally ordered set of *landmark values*, $L_f = \{l_1 < l_2 < \dots < l_k\}$, which must include 0, $f(a)$, $f(b)$, the value of $f(t)$ at every critical point, and may include additional values. It also has a finite totally ordered set of *distinguished time-points*, $T_f = \{a = t_0 < t_1 < \dots < t_n = b\}$, each designating a point where something important happens to the value of f, such as passing a landmark value or reaching an extremum. All functions mentioned from now on should be presumed reasonable.

Definition 4.1 Let $l_1 < \dots < l_k$ be the landmark values of $f : [a, b] \rightarrow \Re^*$. For every $t \in [a, b]$ the *qualitative state of f at t*, $\text{QS}(f, t)$, is a pair $\langle qdir, qval \rangle$, where $qdir$ is 1, 0, or -1 for $f'(t) > 0$, $f'(t) = 0$, or $f'(t) < 0$, respectively, and $qval$ is a landmark value, l_i. The *qualitative state of f on an interval* between two adjacent distinguished time-points, $\text{QS}(f, t_i, t_{i+1})$, is the qualitative state of f at any $t \in (t_i, t_{i+1})$.

Definition 4.2 The *qualitative behavior of f on $[a, b]$*, $\text{QB}(f, [a, b])$, is the sequence of qualitative states $\text{QS}(f, t_0), \text{QS}(f, t_0, t_1), \dots, \text{QS}(f, t_n)$, alternating between states at distinguished time-points and on intervals between adjacent distinguished time-points.

[4] f is continuous on $[a, b]$, continuously differentiable on (a, b), has a finite number of critical values in any interval (i.e., f does not behave pathologically in any interval), and has $f'(a)$ and $f'(b)$ as the left and right limits of $f'(t)$ at a and b, respectively.

Definition 4.3 A *two-terminal system*, $S=\{p_1,p_2\}$, is a pair of functions, each with its set of landmark values, L_{p_i}, and its set of distinguished time-points, T_{p_i}. The set of *distinguished time-points of* S, T_S, is the union $T_{p_1} \cup T_{p_2}$. The *qualitative state of* S at t_i or on (t_i, t_{i+1}), $t_i \in T_S$, is respectively the 2-tuple of states
$\mathrm{QS}(S,t_i)=[\mathrm{QS}(p_1,t_i),\mathrm{QS}(p_2,t_i)]$, or
$\mathrm{QS}(S,t_i,t_{i+1})=[\mathrm{QS}(p_1,t_i,t_{i+1}),\mathrm{QS}(p_2,t_i,t_{i+1})]$.

Definition 4.4 Let l_i and l_j be landmarks of $p_1, p_2 : [a,b] \rightarrow \Re^*$, respectively. l_i and l_j are *corresponding values*, if $\exists t \in [a,b]$ such that $p_1(t)=l_i$ and $p_2(t)=l_j$, where t is called a *corresponding time-point* of p_1 and p_2.

Definition 4.5 The *qualitative behavior of a two-terminal system* S *on* interval $[a,b]$, $\mathrm{QB}(S,[a,b])$, is the sequence $\mathrm{QS}(S,t_0),\mathrm{QS}(S,t_0,t_1),\mathrm{QS}(S,t_1),...,\mathrm{QS}(S,t_n)$, for $t_i \in T_S$.

Definition 4.6 Let $S=\{p_1,p_2\}$ be a system with a set of distinguished time-points T_S. The totally ordered *set of corresponding time-points of* S, T_{S_c}, is the intersection $T_{p_1} \cap T_{p_2}$. It contains the time-points t_0, t_n, and every corresponding distinguished time-point of p_1 and p_2.

4.2 Qualitative Functional Behavior

Given that a two-terminal system in isolation is at some equilibrium state, the qualitative behavior it will exhibit depends on how the input parameter, p_1, will be perturbed. For example, if p_1 is at its landmark l_i $(1 < i < k)$ and it is perturbed to increase (decline), it will go through landmarks $l_{i+1}, l_{i+2}, ..., l_k$ $(l_{i-1}, l_{i-2}, ..., l_1)$. In other words, if we let the system behave only over a subset of its operational regions, it will exhibit only part of the behavior it is capable of. However, if p_1 is at its landmark l_1 and is perturbed to increase, the system will exhibit its behavior over all of its operational regions.[5] We call such a behavior the *qualitative functional behavior* of a two-terminal system.

Definition 4.7 Let $S=\{p_1,p_2\}$ be a two-terminal system and let $l_1, l_2, ..., l_k$ be the landmark values of p_1. The *qualitative functional behavior* of S, $\mathrm{QFB}(S,[a,b])$, is the sequence of states $\mathrm{QS}(S,t_0),\mathrm{QS}(S,t_0,t_1),...,\mathrm{QS}(S,t_{n-1},t_n),\mathrm{QS}(S,t_n)$, where $t_0, t_1, ..., t_n$ are the distinguished corresponding time-points of p_1 and p_2, $p_1(t_0)=l_1$, $p_1(t_\infty)=l_k$, and $qdir(p_1,t)=1$ (i.e., inc) for every $t \in [t_0,t_n]$. $T_{S_F}=\{t_0,t_1,...,t_n\}$ is called the set of *functional time-point* of S.

The qualitative functional behavior of a two-terminal system can be graphically plotted as a two-dimensional piecewise linear function (see Figure 2). Notice that the exclusion of qualitative states at functional time-points from a qualitative functional behavior does not change the shape of such a function. This leads us to define the term *ql-function*.

[5] p_1 will continue to increase to time-point t_∞ because there is no feedback loop that will cause it to change its behavior.

Definition 4.8 Let the qualitative functional behavior of a two-terminal system S be $\mathrm{QFB}(S,[a,b]) = \mathrm{QS}(S,t_0),\mathrm{QS}(S,t_0,t_1),...,\mathrm{QS}(S,t_{n-1},t_n),\mathrm{QS}(S,t_n)$.
The *qualitative piecewise linear function* or *ql-function of* S, QF_S, is the sequence of qualitative states $\mathrm{QS}(S,t_0,t_1)$, $\mathrm{QS}(S,t_1,t_2)$, ..., $\mathrm{QS}(S,t_{n-1},t_n)$. $\mathrm{QF}_S[i]$ is the i-th element in QF_S.

4.3 Qualitative Configuration

A physical system is a collection of disjoint components that are structurally connected in parallel (and/or series). We define one *configuration constraint* called PARALLEL to represent a parallel connection of components. Accordingly, we represent a configuration of a two-terminal system made from components connected in parallel as a collection of PARALLEL constraints on two-terminal components.

Definition 4.9 Let $S_1=\{p_1,p_2\}$ and $S_2=\{p_1,p_2\}$, $p_1, p_2 : [a,b] \rightarrow \Re^*$, be a pair of two-terminal systems that when connected in parallel they produce a new two-terminal system $S_3=\{p_1,p_2\}$. $\mathrm{PARALLEL}(S_1,S_2,S_3)$ is a three-place predicate on two-terminal systems which holds iff $p_2^{S_1}(t)+p_2^{S_2}(t)=p_2^{S_3}(t)$ for every $t \in [a,b]$.

Since a ql-function is simply a short representation of the qualitative functional behavior of a two-terminal component, we can identify conditions under which a PARALLEL constraint can be mapped onto algebraic operations on ql-functions. Before doing so, we define conditions under which two ql-functions are equivalent.

Definition 4.10 Let $S_1=\{p_1,p_2\}$, $S_2=\{p_1,p_2\}$, $S_3=\{p_1,p_2\}$ be two-terminal systems, all with a set of functional time-points, $T_{S_F}=\{t_0,...,t_n\}$. $\mathrm{QF}_{S_1}+\mathrm{QF}_{S_2} \equiv \mathrm{QF}_{S_3}$ iff $\mathrm{QF}_{S_1}[i]+\mathrm{QF}_{S_2}[i] \equiv \mathrm{QF}_{S_3}[i]$ for every i, $1 \le i \le n$. $\mathrm{QF}_{S_1}[i]+\mathrm{QF}_{S_2}[i] \equiv \mathrm{QF}_{S_3}[i]$ iff for adjacent time-points t_{i-1} and t_i in T_{S_F}:

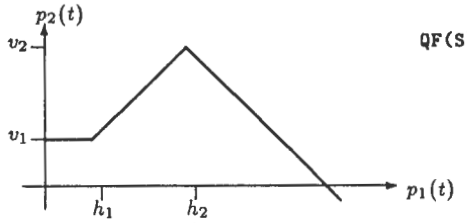
1. $qdir(\mathrm{QS}(p_2^{S_3},t_{i-1},t_i)) = qdir(\mathrm{QS}(p_2^{S_1},t_{i-1},t_i)) + qdir(\mathrm{QS}(p_2^{S_2},t_{i-1},t_i))$, and
2. $qval(\mathrm{QS}(p_1^{S_3},t_{i-1},t_i)) = qval(\mathrm{QS}(p_1^{S_1},t_{i-1},t_i)) = qval(\mathrm{QS}(p_1^{S_2},t_{i-1},t_i))$.

It follows from definitions 4.8, 4.9, and 4.10 that the ql-function of a system S, QF_S, that is configured from two components connected in parallel, S_1 and S_2, is the algebraic sum $\mathrm{QF}_{S_1}+\mathrm{QF}_{S_2}$ of the ql-functions of components S_1 and S_2.

Proposition 4.1 Let $S_1=\{p_1,p_2\}$, $S_2=\{p_1,p_2\}$, $S_3=\{p_1,p_2\}$ be two-terminal systems that all have the same set of functional time-points. $\mathrm{PARALLEL}(S_1,S_2,S_3)$ holds iff $\mathrm{QF}_{S_1}+\mathrm{QF}_{S_2} \equiv \mathrm{QF}_{S_3}$.

5 Qualitative Synthesis

This section presents our qualitative synthesis technique and algorithm QSYN which implements it. Given a goal ql-function for the desired behavior of some system, QSYN constructs all linear combinations of elementary ql-functions that are equivalent to that goal ql-function. Each linear combination is mapped onto a




```
QF(S)=(((t0,t1) (p1 1 ( 0,h1) ) (p2 0 [v1] ))
      ((t1,t2) (p1 1 (h1,h2) ) (p2 1 (v1,v2) ))
      ((t2,t3) (p1 1 (h2,inf)) (p2 -1 (v2,-inf))))
```

Figure 2: ql-function representation of a qualitative functional behavior

set of PARALLEL configuration constraints. Each elementary ql-function in a linear combination is mapped onto the component whose qualitative functional behavior it represents.

Constructing a linear combination of elementary ql-function by comparing the algebraic sum of a linear combination of such ql-functions against a goal ql-function is not always feasible. The elementary and the goal ql-functions often do not have the same set of functional time-points. Thus, one cannot use Definition 4.10 to test for equivalence of the linear combination of elementary ql-functions to the goal ql-function. It is therefore necessary to define heuristic synthesis operators on ql-function which enable handling such cases.

5.1 Synthesis Operators on a ql-function

We shall first illustrate the need for such operators, before we define them. Consider the problem of configuring hedge vehicles. The large number of generic vehicles (elementary components) can be grouped into seven classes, denoted $S_1, S_2, ..., S_7$, based on similarity of functionality. Accordingly, when the large number of payoff-profiles of generic vehicles is described qualitatively, all generic payoff-profiles can be described by seven ql-functions, denoted QF_{S_1} to QF_{S_7} in Figure 3.[6]

Suppose we try to synthesize the "ratio-spread" payoff-profile, denoted QF_G in Figure 4, from QF_{S_3} and QF_{S_4} (as indicated by the example in Section 2.1). These three ql-function look as follows:

```
QF(G) = (((t0,t1) (p1 1 ( 0,h1) ) (p2  0 [v1]     ))
        ((t1,t2) (p1 1 (h1,h2) ) (p2  1 (v1,v2)  ))
        ((t2,t3) (p1 1 (h2,inf)) (p2 -1 (v2,-inf))))
QF(3) = (((t0,t1) (p1 1 (0 ,h1 )) (p2  0 [v1]     ))
        ((t1,t2) (p1 1 (h1,inf)) (p2  1 (v1,inf ))))
QF(4) = (((t0,t1) (p1 1 (0 ,h2 )) (p2  0 [v2]     ))
        ((t1,t2) (p1 1 (h2,inf)) (p2 -2 (minf,v2))))
```

Yet, QF_{S_3}+$QF_{S_4} \not\equiv QF_G$ because the assumption that S_3, S_4, and S_G have the same set of functional time-points is violated (see Definition 3.10). Recall that $S_1, S_2, ..., S_7$ are each representing a whole class of components with the same ql-function. That is, QF_{S_4}, for example, represents a whole class of qualitative functional behaviors whose first element is over an arbitrary p_1-*qval*, $(0, h_i) \in (0, \infty)$, and whose second element's slope is in $(-\infty, 0)$. It is therefore necessary to identify a subclass of components in S_3 and a subclass in S_4 which can be used to configure a class of systems with ql-function

[6] QF_{S_1}, QF_{S_2}, and QF_{S_7} are for the class of non option based vehicles, QF_{S_3} and QF_{S_4} for the class of call option based vehicles, and QF_{S_5} and QF_{S_6} for the class of put option based vehicles.

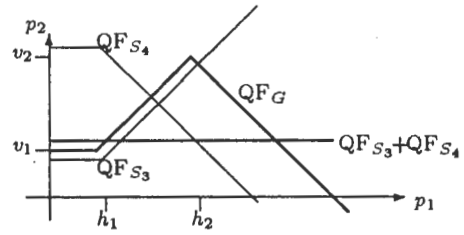


Figure 4: Synthesizing a "ratio-spread" payoff-profile

QF_G. In effect, this translates into a problem of discovering the right ordering of landmark values of QF_{S_3} in relation to those of QF_{S_4}.

5.1.1 Operator *STRETCH*

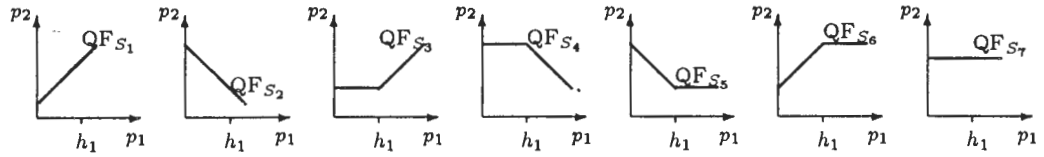
Suppose we attempt to synthesize QF_G from QF_{S_3} and QF_{S_4} (see Figure 5a). $QF_{S_3}[1]$+$QF_{S_4}[1] \equiv QF_G[1]$, but $QF_{S_3}[2]$+$QF_{S_4}[2] \not\equiv QF_G[2]$ because the p_2-*qdir* in $QF_G[2]$ is not equal to the sum of p_2-*qdirs* in $QF_{S_3}[2]$ and $QF_{S_4}[2]$ (see Definition 4.10). The fact that the p_2-*qdir* in $QF_G[2]$ is equal to the sum of p_2-*qdirs* in $QF_{S_3}[2]$ and $QF_{S_4}[1]$, however, suggests that a modified version of QF_{S_4}, denoted QF'_{S_4} in Figure 5b, in which the first element is stretched over the p_1-*qval* $(0, h_2)$, is more likely to contribute to the successful synthesis of QF_G. We can stretch a ql-function element in such a way using operator *STRETCH*.

Definition 5.1 Let $S_i = \{p_1, p_2\}$, $S_j = \{p_1, p_2\}$, and $S_G = \{p_1, p_2\}$ be three two-terminal systems with ql-functions QF_{S_i}, QF_{S_j}, and QF_G, respectively, and let k be an integer $(1 < k \leq n)$ such that:

1. $QF_G[k] \not\equiv QF_{S_i}[k]$+$QF_{S_j}[k]$ because the p_2-*qdir* in $QF_G[k]$ is not equal to the sum of p_2-*qdirs* in $QF_{S_i}[k]$ and $QF_{S_j}[k]$;
2. $QF_G[k-1] \equiv QF_{S_i}[k-1]$+$QF_{S_j}[k-1]$;
3. p_2-*qdir* in $QF_G[k]$ is equal to the sum of p_2-*qdirs* in $QF_{S_i}[k-1]$ and $QF_{S_j}[k]$.

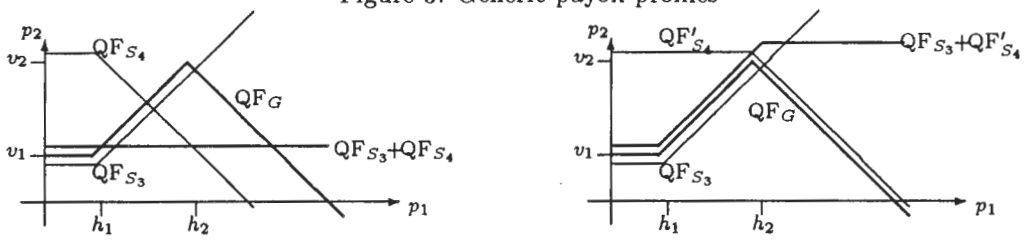
Operator *STRETCH*(QF_{S_i}, k) is used to stretch $QF_{S_i}[k-1]$ over a p_1-*qval* which is the union of the p_1-*qvals* in $QF_G[k-1]$ and $QF_G[k]$.

Operator *STRETCH* actually creates a copy of $QF_{S_i}[k-1]$, changes the p_1-*qval* in that copy to be the p_1-*qval* in $QF_G[k]$, and inserts that copy after $QF_{S_i}[k-1]$ in QF_{S_i}. In the case of a "ratio-spread" ql-function, for example, STRETCH$(QF_{S_4}, 1)$ inserts the new element



p_1 — Parameter modeling the economic variable being hedged
p_2 — Parameter modeling the value of hedged vehicle
h_1 — A landmark representing an arbitrary exercise price

Figure 3: Generic payoff-profiles



(a) Before stretching (b) After stretching

Figure 5: Applying operator *STRETCH* on $QF_{S_4}[1]$

`((p1 1 (h1,h2))(p2 0 [v2]))` right after the first element in QF_{S_4} to create QF'_{S_4} (see Figure 5b). By doing so we achieve $QF_{S_3}[2]+QF'_{S_4}[2] \equiv QF_G[2]$.

5.1.2 Operator *STEEPEN*

Suppose we continue the synthesize of QF_G from QF_{S_3} and QF'_{S_4} (see Figure 6a). $QF_G[3] \not\equiv QF_{S_3}[2]+QF'_{S_4}[3]$ because p_2-*qdir* in $QF_G[3]$ (i.e., -1) is not equal to the sum of p_2-*qdir*s in $QF_{S_3}[2]$ and $QF'_{S_4}[3]$ (i.e., 0). However, if we could modify the p_2-*qdir* in $QF'_{S_4}[3]$ from -1 to -2 to create a new version of QF'_{S_4}, denoted QF''_{S_4} in Figure 6b, we will be able to conclude that $QF_G[3] \equiv QF_{S_3}[2]+QF''_{S_4}[3]$. We can modify the qdir of a ql-function element in such a way using operator *STEEPEN*.

Definition 5.2 Let $S_i = \{p_1, p_2\}$, $S_j = \{p_1, p_2\}$, and $S_G = \{p_1, p_2\}$ be three two-terminal systems with ql-functions QF_{S_i}, QF_{S_j}, and QF_G, respectively, and let k be an integer ($1 \le k \le n$) such that:

1. $QF_G[k] \not\equiv QF_{S_i}[k]+QF_{S_j}[k]$ because the p_2-*qdir* in $QF_G[k]$ is not equal to the sum of p_2-*qdir*s in $QF_{S_i}[k]$ and $QF_{S_j}[k]$;
2. p_2-*qdir* in $QF_{S_i}[k]$ is not zero; and
3. the difference, s, between p_2-*qdir* in $QF_G[k]$ and p_2-*qdir* in $QF_{S_j}[k]$ is not zero.

Operator *STEEPEN*(QF_{S_i}, k, s) is used to increment the p_2-*qdir* of $QF_{S_i}[k]$ by s.

In the case of a "ratio-spread" ql-function, for example, *STEEPEN*$(QF'_{S_4}, 3, -1)$ changes $QF'_{S_4}[3]$ to be `((p1 1 (h2,inf))(p2 -2 (minf,v2)))` to create QF''_{S_4} (see Figure 6b). By doing so we achieve $QF_{S_3}[2]+QF''_{S_4}[3] \equiv QF_G[3]$.

5.2 Algorithm QSYN

Algorithm QSYN is used to synthesize system configurations based on desired behavior. It assumes we know how to check for equivalence of the algebraic sum of two ql-functions to that of another ql-function (see Definition 4.10). It also assumes that we know when it is worthwhile applying operators STRETCH and STEEPEN (see Definitions 5.1 and 5.2),

QSYN receives as input: (1) two reasonable functions p_1 and p_2, each with its totally ordered set of landmarks; (2) a ql-function, QF_G, describing the qualitative functional behavior desired from a prospective system $G = \{p_1, p_2\}$; and (3) a set $\mathcal{QF}$ of n ql-functions, each describing the qualitative functional behavior of a class of elementary components $S_l = \{p_1, p_2\}$, $1 \le l \le n$.

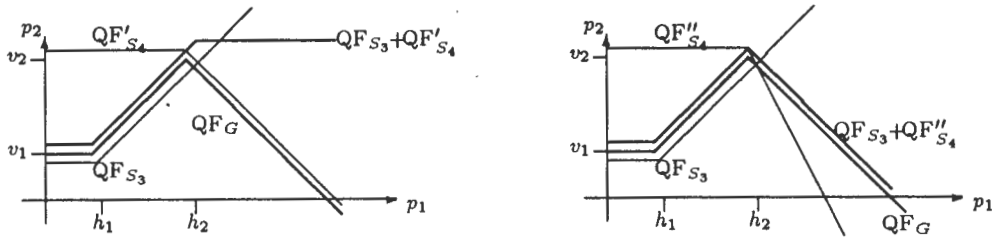
QSYN's output is all possible configurations of system G. A configuration is a set of PARALLEL constraints on components from classes $S_1, S_2, ..., S_n$, where each component is associated with its possibly modified ql-function. A modified ql-function is one on which operator STRETCH and/or STEEPEN was applied. It provides information that characterizes better the components in a specific PARALLEL constraint.

Algorithm QSYN can be summarized as follows:

Step 1: If there is a pair of qfunctions, QF_{S_i} and QF_{S_j} ($i \neq j$), in $\mathcal{QF}$ that has not yet been selected, select it. Otherwise, stop.

Step 2: Check for equivalence of $QF_{S_i}+QF_{S_j} = QF_{S_{ij}}$ to QF_G. If necessary and appropriate apply operator STRETCH on QF_{S_i} and/or QF_{S_j} to discover the right ordering of the p_1 landmarks in QF_{S_i} in relation to p_1 landmarks in QF_{S_j}, and apply operator STEEPEN on QF_{S_i} and/or QF_{S_j} to force consistency of p_2-*qdir*s.

Step 3: If $QF_{S_{ij}}$ matches part of QF_G, add $QF_{S_{ij}}$ to $\mathcal{QF}$, create and store a configuration constraint C_{ij} = PARALLEL(S_i, S_j, S_{ij}) and the possibly modified QF_{S_i} and QF_{S_j} along it. Go to step 5.



(a) Before steepening

(b) After steepening

Figure 6: Applying operator *STEEPEN* on $QF'_{S_4}[3]$

Step 4: If $QF_{S_{ij}}$ matches all of QF_G, print the configuration(s) of S_G you have just synthesized, that is, print: **(a)** constraint C_{ij} and the possibly modified QF_{S_i} and QF_{S_j} for C_{ij}; and **(b)** if S_i (and/or S_j) was itself synthesized from other components by an already stored constraint C_j (and/or C_j), do (a) for C_j (and/or C_j).

Step 5: Go to step 1.

5.3 Output Interpretation

To demonstrate how the output of QSYN is to be interpreted, let us go back to the example from hedging. Suppose QSYN receives as input the goal "ratio-spread" ql-function denoted QF_G in Figure 4, and the ql-functions QF_{S_1}, QF_{S_2},...,QF_{S_7} in Figure 3. After checking for equivalence of QF_{S_4}+QF_{S_4} to QF_G and applying operators STRETCH and STEEPEN on QF_{S_4}, as illustrated in section 5.1 and section 5.2, QSYN produces one configuration constraint PARALLEL(S_3, S_4, G) and stores along it ql-functions QF_{S_3} and QF''_{S_4}. These ql-functions look as follows:

```
QF3  = (((p1 1 (0 ,h1 )) (p2  0 [v1]     ))
        ((p1 1 (h1,inf)) (p2  1 (v1,inf ))))
QF4" = (((p1 1 (0 ,h1 )) (p2  0 [v2]     ))
        ((p1 1 (h1,h2 )) (p2  0 [v2]     ))
        ((p1 1 (h2,inf)) (p2 -2 (minf,v2))))
```

The constraint and these ql-functions tell us that a ratio-spread hedge vehicle can be constructed as follows. QF_{S_3} indicates the purchase of a call option with exercise price h_1 (a generic hedge vehicle from class S_3), whereas QF''_{S_4} indicates the sale of a call option with exercise price h_2 (a generic hedge vehicle from class S_4), where $h_1 < h_2$. Moreover, the absolute value of p_2-*qdir* in $QF''_{S_4}[3]$ indicates the sale of two call options with exercise price h_2.[7]

5.4 Performance Analysis

Algorithm QSYN is computationally tractable, although it searches exhaustively the space of linear combinations of ql-functions to find every combination that produces the goal ql-function. Overall, if n is the number of elementary ql-functions QSYN receives as input, the expected search time of QSYN is $O(n^2)$. However, the qualitative abstraction over the behavior of components with similar functionality usually renders n significantly small. In addition, QSYN's search space is greatly narrowed down by the use of operators STRETCH and STEEPEN. For example, Figure 7 shows part of the search tree for linear combinations of one pair of ql-functions. The emphasized branches in the tree are the ones QSYN chooses to explore. All other branches are pruned by operators STRETCH and STEEPEN.

Our experience indicates that QSYN performs effectively and efficiently. We have implemented QSYN in C++ on a 386-processor IBM-PC. In the case of a goal ql-function with six elements and the elementary ql-functions in Figure 4, for example, QSYN produces all possible configurations of the goal ql-function in less than a second.

6 Scope of Application & Future Work

We have presented a qualitative synthesis technique for the configuration of systems based on some desired behavior. The technique constructs all configurations of a system from two-terminal elementary components connected in parallel. This technique is domain-independent. It requires domain-specific knowledge only for the interpretation of its output.

We are currently applying qualitative synthesis in a prototype expert system called INTELLIGENT-HEDGER, which is used for the design of hedge vehicles [Benaroch, 1992]. We found our technique to be useful for two reasons. One is that it solves the problem for which it is intended well, and the other is that it does not require any pre-stored contextual knowledge such as partial configurations.

Many physical systems, however, involve two complexities which our technique cannot handle. One is that they use components that are connected in series and in parallel to create feedback loops, where some components are not uniform on their input/output parameters. To deal with this complexity, we are now working on the definition of additional configuration constraints to represent structural connections other than parallel ones. We are also modifying QSYN to handle algebraic operations on ql-functions other than addition.

The other complexity is that some physical systems also use multi-terminal components. We are currently exploring one solution approach which looks at the domain of analog computers. Given that the behavior of elementary components in this domain (e.g., deadzone and limit computing components) can be characterized by a ql-function — a qualitative two-dimensional piecewise linear function — [Chorafa, 1965], and that

[7] Which specific call options should be purchased/sold is a matter that is discovered in later design phases not discussed here.

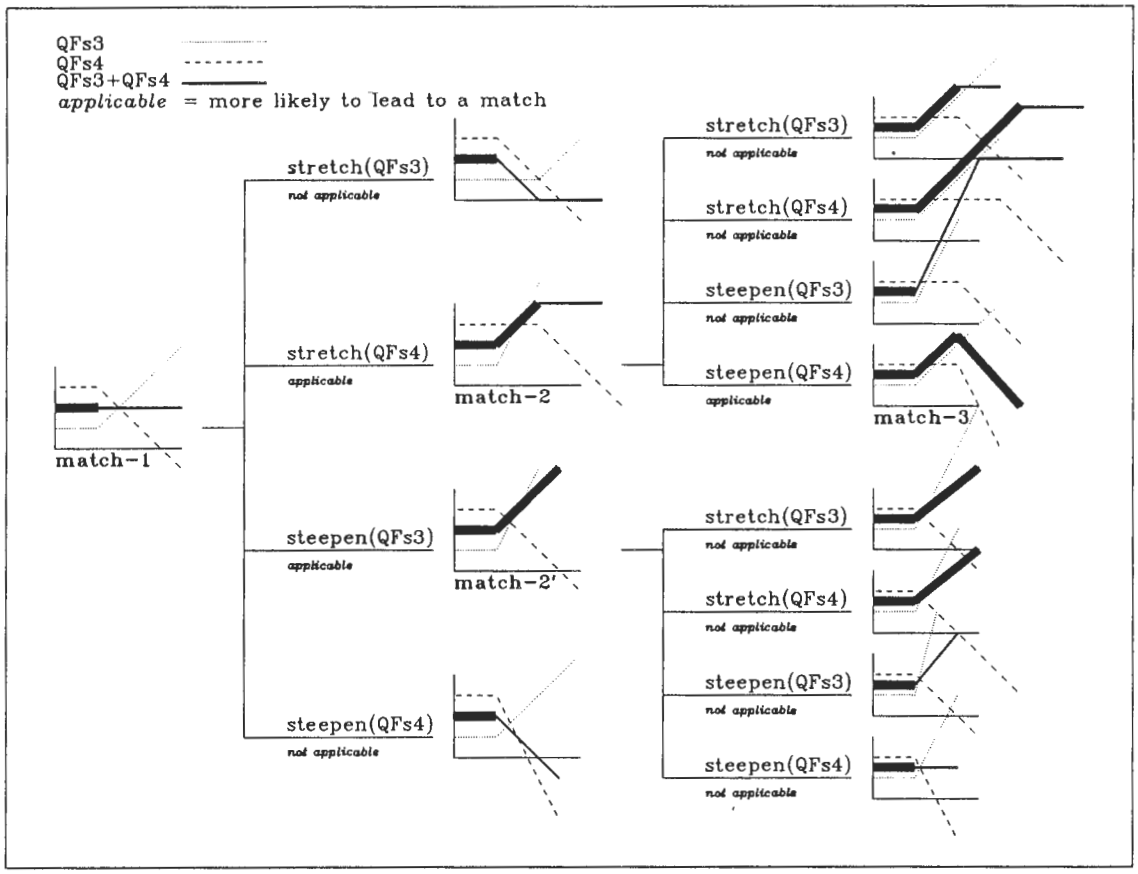


Figure 7: Part of QSYN's search tree for a "ratio-spread" payoff-profile

the desired behavior of some physical system, say a hydraulic servo valve, is also characterized by a ql-function, QSYN can construct analog computer configurations that can simulate that valve's behavior. However, the problem remaining is how to map an analog computer configuration onto a configuration which uses domain-specific components that construct a real hydraulic servo valve. It is not yet clear to us what and how much domain-specific knowledge is needed to carry out such a mapping successfully. In order to be able to answer these questions, we are trying to identify what domain-specific knowledge about the structure of a physical system is being lost when that system is mapped onto an analog computer that can simulate its behavior.

References

[Benaroch, 1992] Benaroch Michel, "Object-Oriented Representations for Reasoning from First-Principles," *Proceedings of HICSS-25*, Hawaii, 1992.

[Benaroch and Dhar, 1991] Benaroch Michel and Dhar Vasant, "An Intelligent Assistant for Financial Hedging," *Proceedings of the 7th IEEE Conference on AI Applications*, Miami, 1991.

[Chorafa, 1965] Chorafa N. Dimitris, *Systems and Simulation*, Academic Press, New York, 1965.

[De Kleer and Brown, 1986] De Kleer J. and Brown J.S., "A Qualitative Physics Based on Confluences," *Artificial Intelligence*, 24:7–83, 1984.

[Elton and Gruber, 1987] Elton J.E. and Gruber J.M., *Modern Portfolio Theory and Investment Analysis (3rd edition)*, John Wiley & Sons, Inc., 1987.

[Hart *et. al.*, 1986] Hart P.E., Barzilay A., and Duda R.O., "Qualitative Reasoning for Financial Assessments: A Prospectus," *AI Magazine*, 7(1):62–68, Winter 1986.

[Kuipers, 1986] Kuipers B., "Qualitative Simulation," *Artificial Intelligence*, 29:289–338, 1986.

[Mittal and Araya, 1986] Mittal Sanjay and Araya Agustin, "A Knowledge-Based Framework for Design," *Proc. of the 5th International Conference on Artificial Intelligence, AAAI-86*, Philadelphia, 1986.

[Saucedo and Schiring, 1968] Saucedo Roberto and Schiring E. Earl, *Introduction to Continuous and Digital Control Systems*, The Macmillan Company, 1968.

An Interactive Constraint-Based Expert Assistant for Music Composition

Russell Ovans* and **Rod Davison†**
Expert Systems Lab
Centre for Systems Science
Simon Fraser University
Burnaby, B.C., Canada V5A 1S6

Abstract

A novel use of constraint propagation within an expert system for music composition is described. The task of composing contrapuntal music is modelled as a constraint satisfaction problem, and consistency techniques are utilized to present the user – as each note is chosen – with a graphical projection of the relaxed constraint graph. The expert system's role is to prevent the user from violating any rule of counterpoint composition. This system illustrates the potential of separating generative (search strategy) from restrictive (constraints) knowledge in interactive expert systems.

1 Introduction

This paper describes an interactive tool for generating first species counterpoint of note against note, a highly structured historical style of music. The tool is essentially a graphical interface to a counterpoint expert system, and would be useful to a beginning student of counterpoint. Our approach is to formulate the task as a constraint satisfaction problem (CSP), and the composition process as the navigation of a constrained search-space. A CSP is the problem of assigning discrete values to a finite set of variables such that a set of constraints is satisfied. The project described here is a natural extension to our earlier work on viewing music composition as a CSP [Ovans, 1990]. We view the composition process as the selection of note attributes (pitch, duration, etc.), from finite and discrete domains, such that a set of constraints is satisfied. In this context, the constraints impose acceptable structure, coherence, and aesthetic on a composition.

The user's task is to compose a counterpoint for a given melody. The user, as composer, is responsible for generating a musical solution whereas the expert system is responsible for policing the constraints.

*The financial support of PRECARN Associates, Toronto, is gratefully acknowledged. E-mail: ovans@cs.sfu.ca.

†The financial support of the Science Council of B.C. is gratefully acknowledged. Author's present address: Vertigo Technology Inc.; 301-1134 Homer St.; Vancouver, B.C.; V6B 2X6.

We begin in Section 2 of this paper with a description of the tool from the user's perspective. The underlying theory of CSPs and a description of how the composition task is formulated as one is found in Section 3. In Section 4 we provide some detail about the actual implementation, and conclude in Section 5 with a short discussion.

2 Expert-Assisted Counterpoint Composition

Counterpoint is a style of music composition that predates our modern notion of harmony. Whereas harmony is concerned mainly with vertical relationships among notes, chords in succession, and supporting a dominant melodic theme, counterpoint is *polyphonic* composition: the combination of several independent and equally interesting melodies into a coherent whole. The rules of counterpoint, which constrain the allowable note combinations occurring in a composition, were codified in 1725 by Johann Fux, a translation of which appears in [Mann, 1965]. The normal manner of composition in counterpoint is to craft additional melodic lines (called *counterpoints*) to be sung, or played, along with a previously composed given melody, usually taken from a book of chorales. The new voices are said to be in counterpoint to the given melody.

The program interface is presented in Figure 1. The window is titled "Fux Composition Tutor" in deference to Johann Fux. The user has a choice of composing a counterpoint of four, seven, or twelve bars in length. In the example, *12 Bar* has been chosen and a given melody automatically generated. The rules of counterpoint are such that a given melody completely defines a search-space of permissible counterpoints. The corresponding counterpoint choices are presented as in the figure: the whole notes in each bar of the counterpoint represent allowable choices, not chords. The user's task is to make choices within the counterpoint search-space until a single note remains in each bar.

There are 5,930 "correct" counterpoints for the given melody presented in Figure 1 [Ovans, 1992]. The expert system's role is to insure the user finds one; that is, that he or she does not violate any of the rules of counterpoint. The task is made interesting by the simple fact that not every option is consistent with every other: user

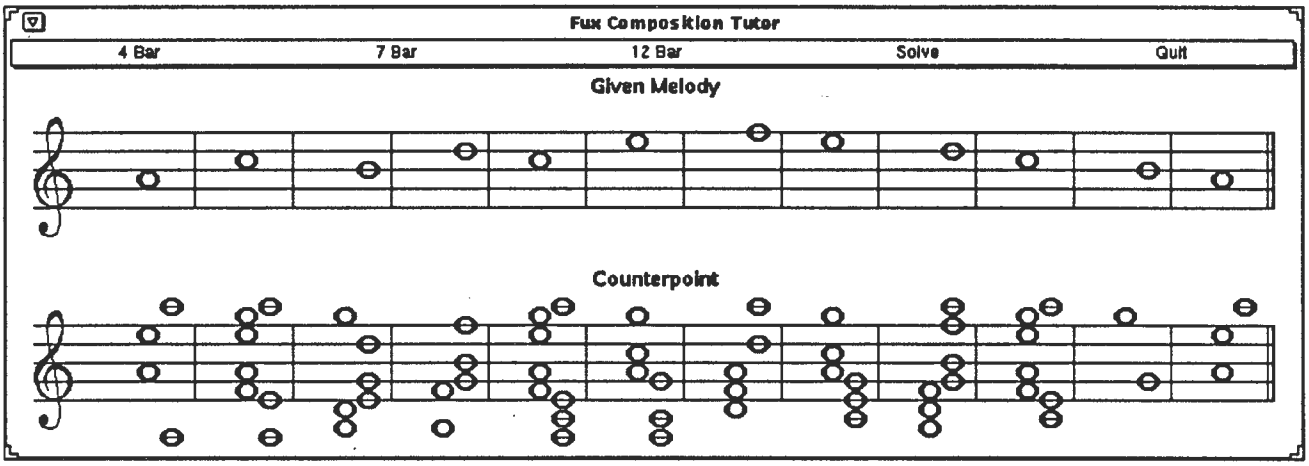


Figure 1: A 12-bar given melody and its resulting counterpoint search-space.

choices result in a narrowing of the search-space.

The search-space is navigated by selecting – with a mouse – a series of notes, which need not be done in the conventional and sequential left-to-right manner. With each selection the expert system propagates constraints and filters the remaining consistent choices in the other bars. Figure 2 reveals the resulting search-space after the user has selected a G in the fifth bar. Because consistency is maintained as each variable is instantiated, the user can make a choice for a pitch value and see the ramifications in the form of the resulting constraint propagation (e.g., "...if I choose a fifth for the interval in this bar, I see that it rules out the A in the previous bar.").

User choices can be undone by selecting a note a second time. Figure 3 is the state of the composition after two subsequent choices are made and the original choice of Figure 2 is repealed.

As well, the user need not finish the task: at any time the expert system can be asked to solve the problem. It does so without overriding the user: previous choices must appear in the solution (see Figure 4).

3 The Underlying Theory

Our approach is unique in its recognition that composing counterpoint can be represented as a constraint satisfaction problem: a user of this tutoring system is interacting with a constraint graph. In this section we present a short summary of the topic of CSPs and show how the task of composing first species counterpoint can be formulated as a CSP.

3.1 Constraint Satisfaction Problems

A ubiquitous problem in artificial intelligence is the constraint satisfaction problem, which can be succinctly stated as follows: given a finite set of variables $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ whose elements range respectively over the finite (and not necessarily numeric) domains $D_1, D_2, \ldots, D_n$, find a value for each variable such that a finite set of constraints is satisfied. The role of the constraints is to reduce the cartesian solution space $\mathcal{D} = D_1 \times D_2 \times \cdots \times D_n$. Each constraint is a relation on a subset of $\mathcal{X}$ that states which values are consistent with each other. CSPs are solved by finding one (all) point(s) in the finite discrete space $\mathcal{D}$ that simultaneously satisfies the constraints. A good survey of CSPs is found in [Mackworth, 1992].

CSPs are conveniently represented by constraint graphs [Mackworth, 1977]. In a constraint graph, each node is a variable to be assigned a value. Adjacent nodes must satisfy the constraint denoted by the arc connecting them. Unary constraints are expressed with loops. The constraint graph for a corresponding graph colouring problem is in fact the graph to be coloured. To represent k-ary constraints, $k > 2$, hyperarcs (arcs that connect more than two nodes) are required.

Depth-first chronological backtracking search is a general purpose algorithm for solving CSPs. Despite the elimination of subspaces from the solution space $\mathcal{D}$ with each failed instantiation, in the worst case backtracking is exponential in the number of variables. Attempts at improving the performance of backtracking algorithms led to the incorporation of consistency techniques. Consistency techniques prune the search-space before failure occurs thus improving the efficiency of tree search by reducing the number of backtrack points. The search-space is reduced by decreasing the tree's branching factor; values from a variable's domain that cannot possibly participate in any solution to the problem are eliminated, thus avoiding a possibly costly failure later on in the search tree.

Consistency techniques are weak inference methods: they always work, but will not always solve a CSP. Arc consistency [Mackworth, 1977] is a technique applicable to connected nodes in a constraint graph. Assuming a binary constraint P is acting on variables x_i and x_j, the arc is consistent *iff*

$$(\forall a \in D_i)(\exists b \in D_j)P(a,b)$$

$$(\forall b \in D_j)(\exists a \in D_i)P(a,b)$$

Note that arc consistency is generalizable to hyperarcs as well, in which case we call it k-ary arc consistency.

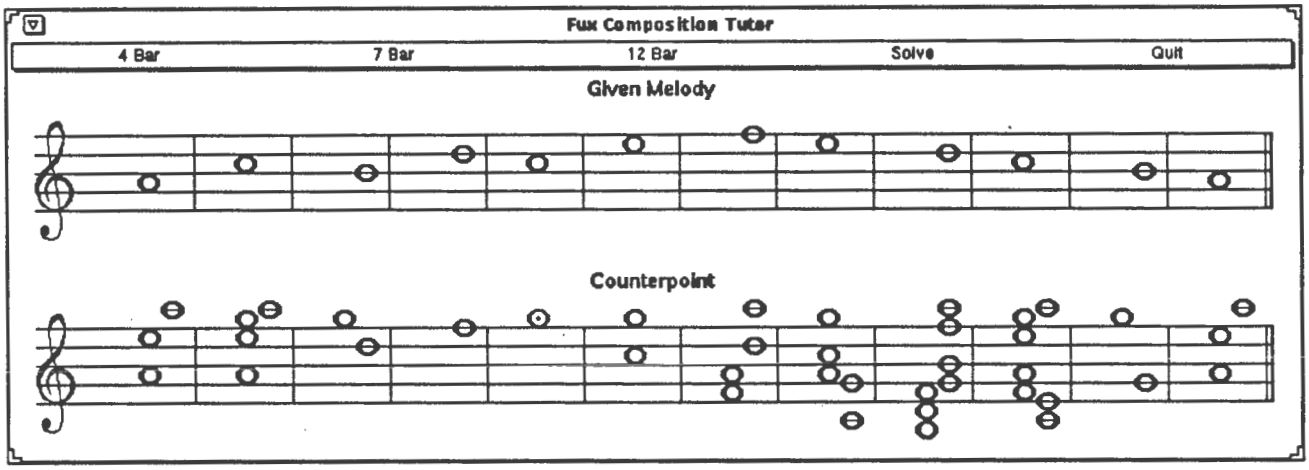


Figure 2: A choice is made resulting in a reduced search-space.

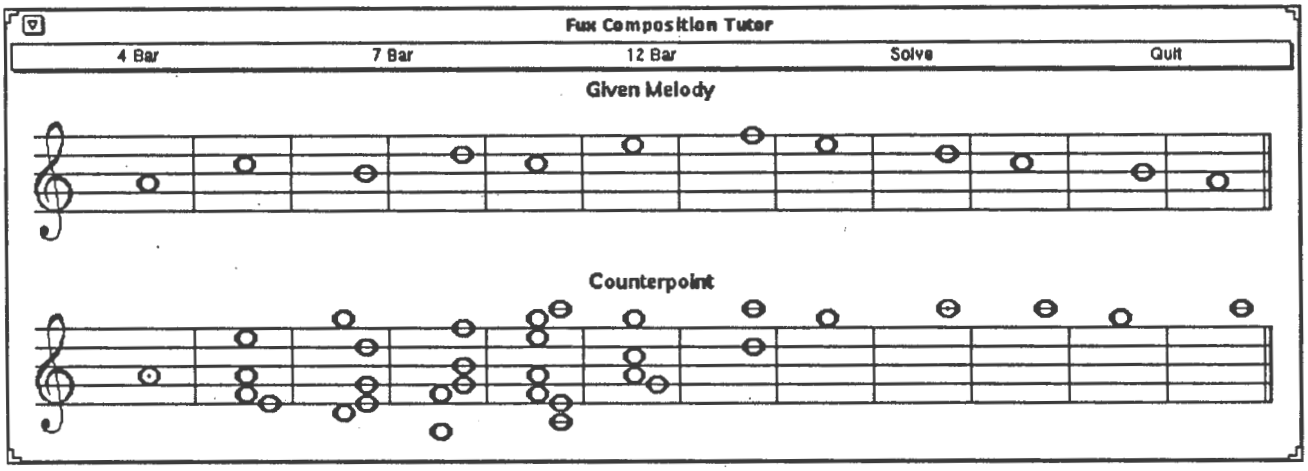


Figure 3: Two subsequent choices plus the original choice revoked.

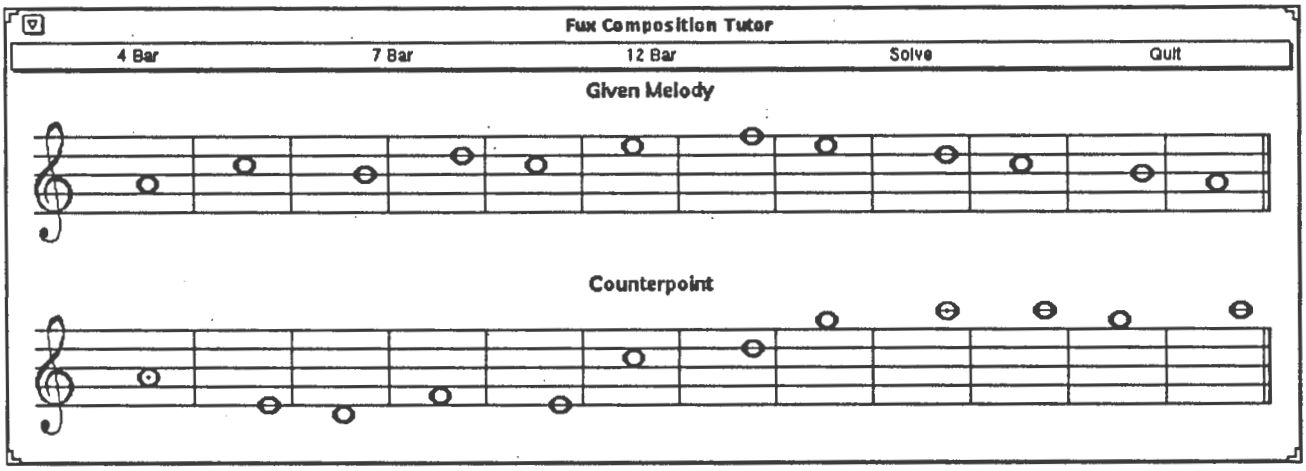


Figure 4: The composition as completed by the expert system.

A constraint graph is said to be arc consistent *iff* all adjacent nodes are consistent with each other. Arc consistency may occur fortuitously, but is assured when a graph is subjected to a full arc consistency algorithm, which in any of its forms essentially controls the propagation of constraints until quiescence. Many polynomial-time arc consistency algorithms have been reported, for example Waltz's procedure [Waltz, 1975] and AC3 [Mackworth, 1977]. A unified survey of many of the CSP-solving algorithms in terms of tree search and arc consistency is given in [Nadel, 1989].

3.2 Counterpoint as a CSP

> *The problem of making a good piece of music is a problem of finding a structure that satisfies a lot of different constraints.* Marvin Minsky [Roads, 1980]

This section describes how the problem of generating contrapuntal music can be formulated as a CSP. Specifically, compositions of first species counterpoint that adhere to a fairly complete set of rules taken from [Mann, 1965] are modelled. The first species is counterpoint of the simplest form: two or more voices comprised of notes of equal length. This restriction to first species eliminates the myriad of representation problems concerned with rhythm and allows us to focus solely on the constraint relationships. The only attribute of a note that requires representation is its pitch. Any representation scheme that facilitates the definition of constraints on pitch values would be acceptable. We have chosen the Musical Instrument Digital Interface (MIDI) standard: pitch is an integer from 1 to 127 corresponding to an ordering of the semi-tones with middle-C equal to 60.

A first species counterpoint composition in two voices n bars in length is thus comprised of n counterpoint variables $\{c_1, \ldots, c_n\}$ and n melody variables $\{m_1, \ldots, m_n\}$. It is convenient to think of the notes of the given melody as instantiated variables rather than constants. Using this notation, the ith bar of a composition is comprised of note m_i in the melody and c_i in the counterpoint. The task is thus to assign values to $\{c_1, \ldots, c_n\}$ that satisfy the rules of first species counterpoint given an instantiation for each of $\{m_1, \ldots, m_n\}$.

First species counterpoint, as codified in [Mann, 1965], is defined by nine rules restricting allowable note combinations. Each of these rules can be represented as a set of local constraints enforced by the same predicate [Ovans, 1992]; in other words, each rule is comprised of many constraints. A sample rule is the following: because they are difficult to sing, melodic intervals greater than a minor sixth or equal to a tritone are forbidden. A constraint defining allowable melodic intervals is thus:

$$\text{melodic}(c_i, c_{i+1}) \Leftrightarrow \mid c_i - c_{i+1} \mid \leq 8 \wedge \mid c_i - c_{i+1} \mid \neq 6$$

This constraint is applied to successive neighbouring pairs of counterpoint variables. Note that this constraint would normally also apply to the melody variables, but given that our system works only with supplied (and correct) melodies, constraints acting solely on melody variables are superfluous.

A constraint graph that enforces the nine rules of note against note first species counterpoint is presented in Figure 5. This graph is for compositions of length four – the shortest compositions to include all the rules of first species counterpoint – and can easily be extended for compositions of greater length. It is in actuality a hypergraph: the *skip-step* constraints are ternary, while the *quaternary* constraints embody both the forbidden parallel motion to a perfect consonance and forbidden progressions to an octave by a skip rules.

For clarity, unary constraints are not included in the graph of Figure 5. The unary constraints restrict the pitch values for the notes of the counterpoint to those of the Aeolian mode since given melodies are assumed to be in the Aeolian mode. This provides a convenient mechanism for prohibiting a change of key. Since counterpoint is generally written for voice, we impose an arbitrary two-octave range:

$$\text{mode}(c_i) \Leftrightarrow c_i \in \{45, 47, 48, 50, 52, 53, 55, 57, 59, 60, 62, 64, 65, 67, 69\}$$

The *mode* constraint is applied to each counterpoint variable, save the second to last. To facilitate the formation of the proper cadence, the unary predicate *cadence* is applied to the second to last counterpoint variable:

$$\text{cadence}(c_{n-1}) \Leftrightarrow c_{n-1} \in \{56, 68\}$$

With an instantiated set of melody variables and the domains of the counterpoint variables initialized to reflect these unary constraints, a full arc consistency algorithm applied to the constraint graph results in a drastically reduced branching factor for the search tree of permissible counterpoints [Ovans, 1992]. It is a projection of this resulting relaxed constraint graph that is presented to the user in Figure 1.

4 The Implementation

The interface is a C++ program that utilizes the InterViews user interface toolkit. The expert system is an Echidna knowledge-base. Echidna is a new constraint logic programming language – strongly influenced by CHIP [Van Hentenryck, 1989] – suitable for model-based expert systems applications. Echidna improves on existing expert system shells by combining facets of object-oriented programming, dataflow dependency backtracking, and constraint logic programming [Havens *et al.*, 1990].

The Echidna knowledge-base consists of a set of schema (object class) declarations, including those for *composition*, *counterpointNote*, and *melodyNote* classes. The constraints of counterpoint are realized as methods (Horn clauses) within these schema declarations. Bi-directional constraints between schema instances (objects) are created via message passing. The Echidna reasoning engine maintains k-ary arc consistency on all discrete constraints. Echidna's dependency-directed backtracking provides efficient means for undoing user selections.

The graphical interface and Echidna interpreter are separate processes. The two processes communicate

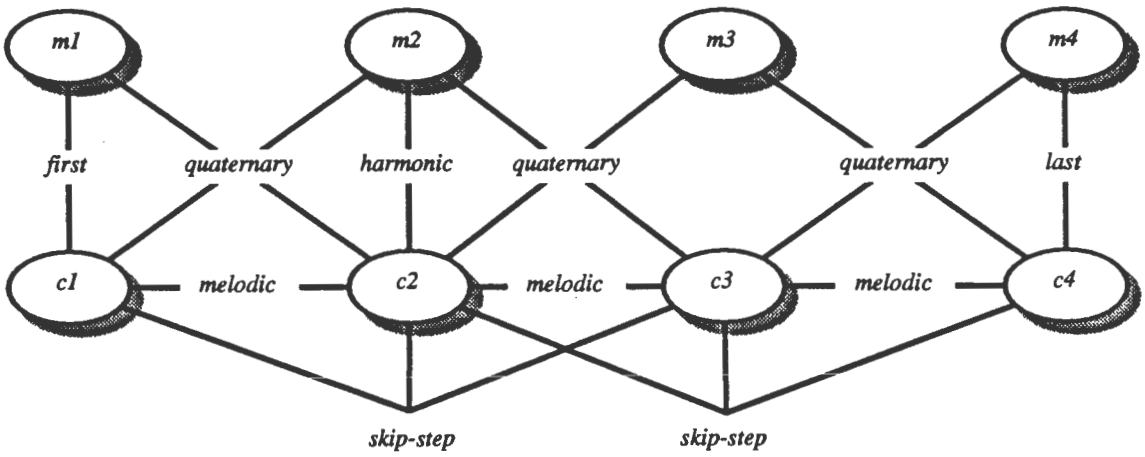


Figure 5: A constraint graph for first species counterpoint, $n = 4$.

through a Unix pipe, and need not reside on the same CPU.[1] The protocol governing their communication is defined by Echidna's external object protocol. For a process external to Echidna, this protocol provides the ability to issue goals to the knowledge-base and "share" logic variables. In our composition system, the pitch value for each *counterpointNote* instance is shared with the interface process. Any change to the domain of a shared variable is communicated from the interface to Echidna (in the case of a user choice), and vice-versa (in the case of constraint propagation).

5 Discussion

Rule-based automated music composition systems (like that of [Ebcioglu, 1984; Thomas, 1985]) typically contain a set of constraints that eliminate unmusical compositions. The problem with these systems is two-fold. First, passive constraints are computationally expensive. The incorporation of consistency techniques can help to ameliorate inefficient generation of compositions due to excessive backtracking [Ovans, 1992], but they are by no means a panacea.

Second, composition is not paint-by-numbers. In other words, the satisfaction of constraints alone is not sufficient for musical results [Ebcioglu, 1984]. The composer's skill is in how he or she prunes and navigates the constrained, but still very large, search-space of possibilities. How to represent this skill in a declarative knowledge base is not well understood and indeed might be impossible for deep humanistic domains like music [Loy, 1991].

The problem, as it also appears in other expert systems, is as follows. Knowledge of how to generate a "good" solution is difficult to represent. Conversely, constraints that restrict allowable solutions are often easy to represent. This dichotomy is reinforced by the experiences of others who, in an attempt to devise rules for finding good solutions, have ended up with systems that contain hundreds of rules yet generate mediocre compositions (e.g., [Schottstaedt, 1984; Ebcioglu, 1984]). The design of our composition tutor is influenced by the belief that, in this case, the user best provides the generative knowledge. Let the expert system represent and propagate constraints; this it can do well and efficiently. Let the user search for the good solutions for this he or she (currently) does better than the expert system. The constraint-based approach is useful in this domain and others like it (e.g., interactive scheduling) where it is more important for the computer to prevent mistakes than to generate solutions.

[1] We normally run the Echidna process on a NeXTstation and the interface under OpenWindows on a SPARCstation.

References

[Ebcioglu, 1984] Kemal Ebcioglu. An expert system for schenkerian synthesis of chorales in the style of J. S. Bach. In *Proceedings of the International Computer Music Conference*, pages 135–142, 1984.

[Havens *et al.*, 1990] William S. Havens, Susan Sidebottom, Greg Sidebottom, John Jones, Miron Cuperman, and Rod Davison. Echidna constraint reasoning system: Next-generation expert system technology. Technical Report CSS-IS TR 90-09, Centre for Systems Science, Simon Fraser University, 1990.

[Loy, 1991] D. Gareth Loy. Connectionism and musiconomy. In *Proceedings of the International Computer Music Conference*, pages 364–374, 1991.

[Mackworth, 1977] Alan K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[Mackworth, 1992] Alan K. Mackworth. Constraint satisfaction. In Stuart C. Shapiro, editor, *Encyclopedia of Artificial Intelligence, Second Edition*, pages 285–293. John Wiley & Sons, New York, 1992.

[Mann, 1965] Alfred Mann, editor. *The Study of Counterpoint from Johann Joseph Fux's Gradus Ad Parnassum*. W. W. Norton, New York, 1965.

[Nadel, 1989] Bernard A. Nadel. Constraint satisfaction algorithms. *Computational Intelligence*, 5(4):188–224, 1989.

[Ovans, 1990] Russell Ovans. Music composition as a constraint satisfaction problem. In *Proceedings of the International Computer Music Conference*, pages 317–319, 1990.

[Ovans, 1992] Russell Ovans. Efficient music composition via consistency techniques. Technical Report CSS-IS TR 92-02, Centre for Systems Science, Simon Fraser University, 1992.

[Roads, 1980] Curtis Roads. Interview with Marvin Minsky. *Computer Music Journal*, 4(3):25–39, 1980.

[Schottstaedt, 1984] B. Schottstaedt. Automatic species counterpoint. Technical Report STAN-M-19, Centre for Computer Research in Music and Acoustics, Stanford University, 1984.

[Thomas, 1985] Marilyn Taft Thomas. Vivace: A rule based AI system for composition. In *Proceedings of the International Computer Music Conference*, pages 267–274, 1985.

[Van Hentenryck, 1989] Pascal Van Hentenryck. *Constraint Satisfaction in Logic Programming.* MIT Press, Cambridge, Massachusetts, 1989.

[Waltz, 1975] David Waltz. Understanding line drawings of scenes with shadows. In Patrick Henry Winston, editor, *The Psychology of Computer Vision.* McGraw-Hill, New York, 1975.

A Customization Environment for the Expert Advisor Network Management System

Tony White
Computer Science Department
Carleton University
Ottawa, Ontario K1S 5B6
email: twhite@turing.scs.carleton.ca

Andrzej Bieszczad
Bell Northern Research
Ottawa, Ontario K1Y 4H7
email: andrzej@bnr.ca

Abstract

Expert Systems have become one of the key innovative technologies applied to network surveillance. The most important part of the expert system is the knowledge base that it uses to detect network faults and generate troubleshooting advice. It is very difficult to provide a generic knowledge base that covers all the requirements of end users, hence customization becomes important. This paper describes the Expert Advisor[1] customization environment that provides a rich, symbolic knowledge encoding environment for the expert system. Firstly, an overview of the network surveillance system is presented. It is followed by an overview of the knowledge representation that is used as the encoding mechanism for network problems. The editing, compilation and off-line verification cycles are explained. The verification methods at three verification levels are introduced. The paper concludes with a description of the benefits of customization in the context of network surveillance expert systems.

1. Introduction

The Expert Advisor is an expert system designed to monitor and diagnose faults that can occur in a packet switching network. See [White, Bieszczad 1992] for a detailed description of the Expert Advisor. It introduces the concept of a *problem.*

[1] Copyright Northern Telecom

The knowledge representation used in the Expert Advisor is a hybrid of model [Kahn, Kepner and Pepper 1987] and rule [Laffey, et al 1988] based approaches. The knowledge base supports a fault model that consists of a number of problem descriptions that are maintained in ASCII files. The problem scripts are written in an English-like language called the Problem Description Language (PDL). Users modify or create problem scripts with the help of a customization environment. These files are compiled into record structures that are subsequently used by the expert system.

The knowledge base can be altered on-line without interrupting the monitoring of the network. This implies that the expert system can work with multiple knowledge bases and can switch from one to another on-line. Additionally, the modularity of the knowledge base allows the system administrator to alter only selected problem scripts in the active knowledge base.

2. Knowledge Representation Overview

Each problem description is a frame that has several slots containing filters that define the network events that signal the start of the problem, the events that are part of the problem once activated and a data base query used to retrieve relevant events from the event archive. These filters are applied to all the events that are generated by the corresponding network component. All events passed by the filters are added to the set of events that constitute the problem. The dynamic behaviour of the problem is

encoded in a set of forward chaining rules. Rules have the form **If** *conditions* **Then** *actions*, where *conditions* test various aspects of the network event data and *actions* are taken from a set of system defined functions. The defined functions provide for assignment, data base query, interaction with other processes and numerous other facilities. The rules, in effect, specify which events change the state of the problem.

In addition to an event set associated with a problem, a set of facts (or states) may be associated with each problem. A fact can be created or deleted by the actions of rules.

Facts can be used, for example, to store the current state of the component or historical information such as the number of times it has failed.

Problems can be related to other problems and information can flow from one to another via bi-directional communication channels. These channels are created dynamically as new problem instances are created, or existing ones destroyed. The exchange of information via these communication (or message) channels has allowed us to generate a very large, modular knowledge base (10,000 rules) which has proven to be maintainable.

2.1 Knowledge Base Maintenance Overview

As a result of the dynamic nature of modern day networks, it is almost a certainty that new problems will arise, some problems will need to be corrected and the nature of others will change. Over time, the network surveillance knowledge base may degrade and the advisory capabilities of the expert system may also degrade if left unchanged. The end user must be provided with a method of maintaining the knowledge base - the need for a customization environment becomes evident. Customization thus ensures an improving level of expertise will be demonstrated by the expert system as the network evolves.

Such an environment should be symbolic in nature, providing a *what-you-see-is-what-you-get* interface and should provide a revision control capability in order that a historical record of knowledge base modifications can be maintained. The customization environment should provide a mechanism for *verifying* changes to the knowledge base before the changes are introduced into the live network. Hence, it should operate off-line and on the same workstation as the live expert system. Figure 1 demonstrates this point graphically - a single workstation can be used to run both the off line simulation and the online real time environments. In the former, the event source is a file, in the latter the source is the network itself.

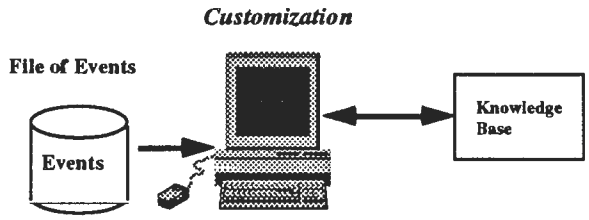


Figure 1.

2.2 Functionality of the Customization Environment

There are two main loops involved in the knowledge extraction process: the *knowledge encoding* and *simulation* cycles. These two cycles are shown in figure 2. In the knowledge encoding cycle, the user may modify the knowledge base using the editing capabilities of the customization environment. Any of the problem scripts already in the knowledge base can be modified and new problem descriptions can be added. The changes to the knowledge base are recorded by the script revision control system. Using the PDL compiler, the user ensures that a new or modified script is syntactically correct. If a script proves to be syntactically incorrect, the compiler stops and the appropriate script is reloaded in the customization environment, highlighted at the offending line. The error messages reported by the compiler are then used as a guide by the user on how to encode knowledge in the script correctly.

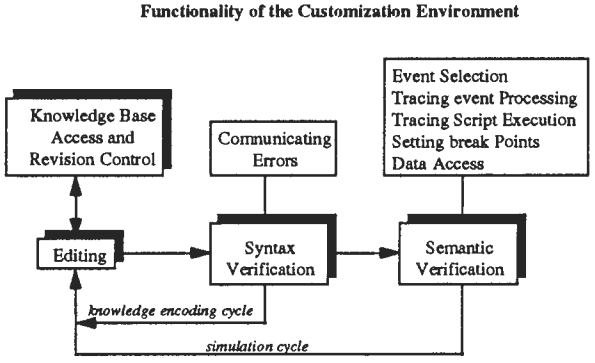


Figure 2.

In the simulation cycle, the compiled scripts may be loaded into the simulator and be verified against a set of network events collected in a file by the on-line expert system. Subsets of the stored events may be chosen for processing. The analysis of the stream of events can be traced using the control capabilities of the simulator. The state of problem instances may be observed using the data tracing features.

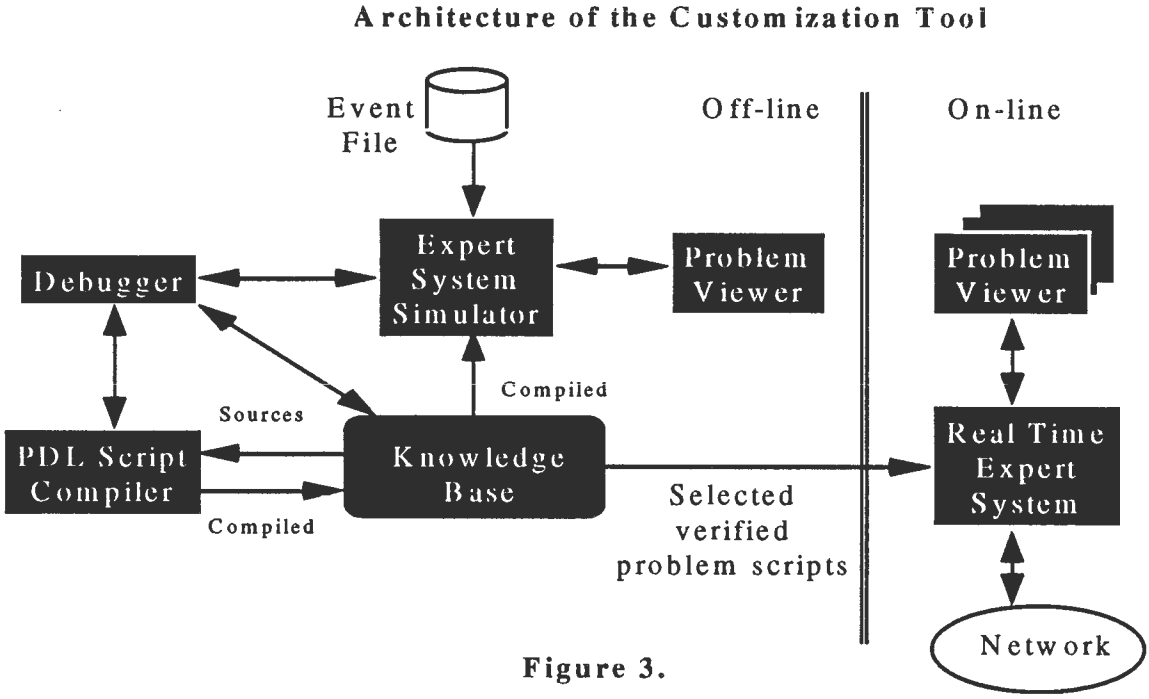


Figure 3.

At any time, the user may make further modifications to the script to fix syntax errors (knowledge encoding cycle) or to fix any semantic errors (simulation cycle). Semantic errors may take the form of incorrect messages being displayed, events being incorrectly associated with a given problem or the hierarchy of problems being incorrectly calculated.

2.3 Architecture of the Customization Environment

There are several components to the customization environment. They are shown in figure 3.

- The *simulator* behaves like the real time expert system, but reads events from a file and can be suspended at a particular event, rule, action or state change. The state of the suspended simulator can be interrogated, so that the processing of network events can be observed.

- The *debugger* provides the user with an integrated knowledge base customization environment including problem script management, editing and compiling capabilities as well as a user interface used to control the simulator.

- The *PDL script compiler* is the task that ensures the syntactic integrity of the script and translates the script into a concise internal format understood by the simulator (or real time) systems. A number of optimizations are performed during the compilation process that significantly reduces the number of rules and filters that are evaluated when a network event is processed.

- The *problem viewer* is the tool primarily used on-line to access the data base of network problems. It can be used with the simulator in order to test the results of the simulated event analysis and ensure that the ultimate user - the network operator - is presented with correct information.

The customization tool provides a *what-you-see-is-what-you-get* environment. All tools that interact with the simulator behave in the same way as they do when actually monitoring the network. Hence the changes made to a knowledge base when exercised during a simulation will cause the problem viewer displays to react in the same way as would occur when executing in real time.

2.4 Knowledge Base Verification Methodology

There are three levels of verification that a problem script must pass. These are: event processing, rule evaluation and inter-problem communication verification levels. These levels are described in the next three sub sections.

Event processing

The event processing phase ensures that all relevant network events are handled and that the expected problem instances are being correctly reported in the problem viewer and associated surveillance tools. If the results are satisfactory, the next two verification phases can be

skipped as, to a large extent, they are encapsulated in this one. During this phase a number of things are exercised:

- Selecting events for a simulation. Including and excluding network events can be performed.
- Testing whether a specific event or event sequence generates the desired network problem.
- Use of the problem viewer to detect whether the problem instance is correctly displayed and that message and network event browsers contain the correct information.

Rule evaluation

The user may set watch points or break points within a specific script in order to determine whether particular execution paths are being exercised. The system reports all visited watch points without suspending the processing of network events. At a break point, the simulation is suspended and the current state of the simulation can be interrogated or modified using specialized debugger browsers. The state of the simulation can subsequently be restored, if required, and the remaining events processed. During this phase a number of things are exercised:

- Observing that the correct rules are evaluated upon reception of a state change from a problem instance or network event.
- Verifying that the actions of the firing rules are executing properly.
- Verifying that problem instances behave properly and the browsers in the problem viewer present correct information.
- Using watch and break points along with simulation stepping features to follow network event evaluation.

Inter-problem communication

Problem instances communicate with each other by sending and receiving state change information. The user can manually change, or create, state changes by modifying the value of a fact. In this way, state changes propagate through the communication channels between problem instances to other problem instances. Watch and break points may be used to verify the correctness of the propagation process. During this phase a number of things are exercised:

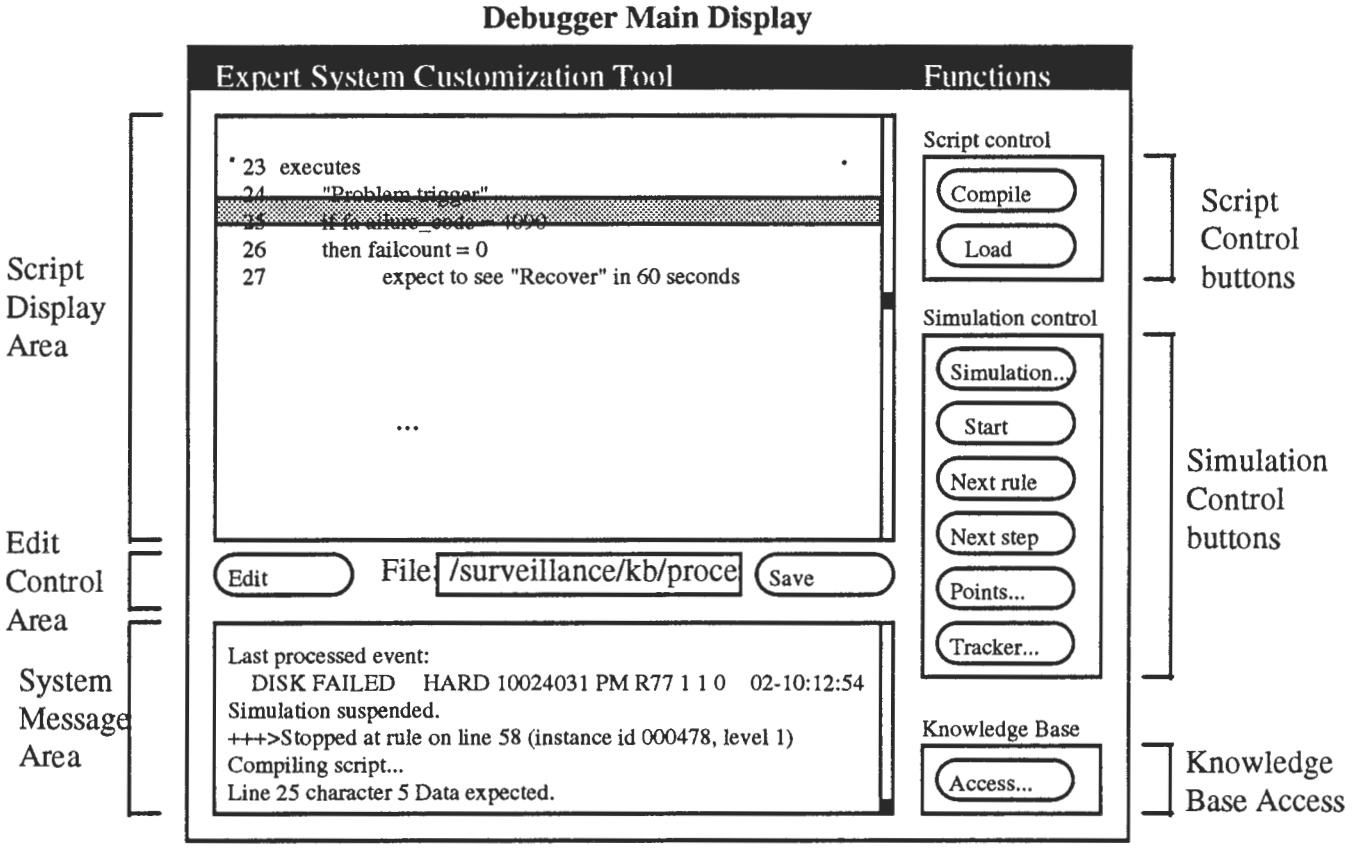


Figure 4.

- Observing that communication between problem instances can take place by stepping through the rules processed during the evaluation of a network event.

- Verifying that the fact values, when changed, are correctly propagated to other problem instances.

- Using the problem viewer to observe that the correct problem hierarchy is generated and maintained.

Frequently, the channels between different problem scripts will be disabled while debugging of the rules that relate directly to the network event occurs. In these cases, state changes are manually propagated to the appropriate problem instances. Hence the knowledge base can be tested in a modular way. The dynamic behaviour of individual problem scripts can be tested, the events associated with problem scripts can be tested, and finally the interfaces between problems can be tested. Also, all browsers available in the real-time system can be used to ensure that the correct displays are generated (for example, correct state information is reported).

3. Debugger User Interface

The debugger part of the customization environment has been designed to provide a state of the art, intuitive, symbolic debugging environment. An intuitive user interface is the key to the effective use of the knowledge base customization tool. Figure 4 shows the Main Display of the debugger tool. The debugger provides the following features:

- Knowledge base maintenance facilities. The debugger uses a revision control system in order to log all changes that take place to the knowledge base being customized. Being directory based, changing from one knowledge base to another during the simulation process is simply accomplished. Multiple knowledge bases can be tested during a single simulation session.

- Save and restore the state of the simulator. The instantaneous state of the simulator can be stored on disk, and recovered for further analysis. In this way, knowledge base changes can be made, the effects evaluated and subsequently discarded before proceeding with a previous session.

- Script editing. A full screen editor, with search and replace capabilities, is provided. An interface to the PDL compiler is also provided.

- Simulation control facilities. The simulation event file can be selected, or changed. Events can be included or excluded from the simulation. The simulation speed

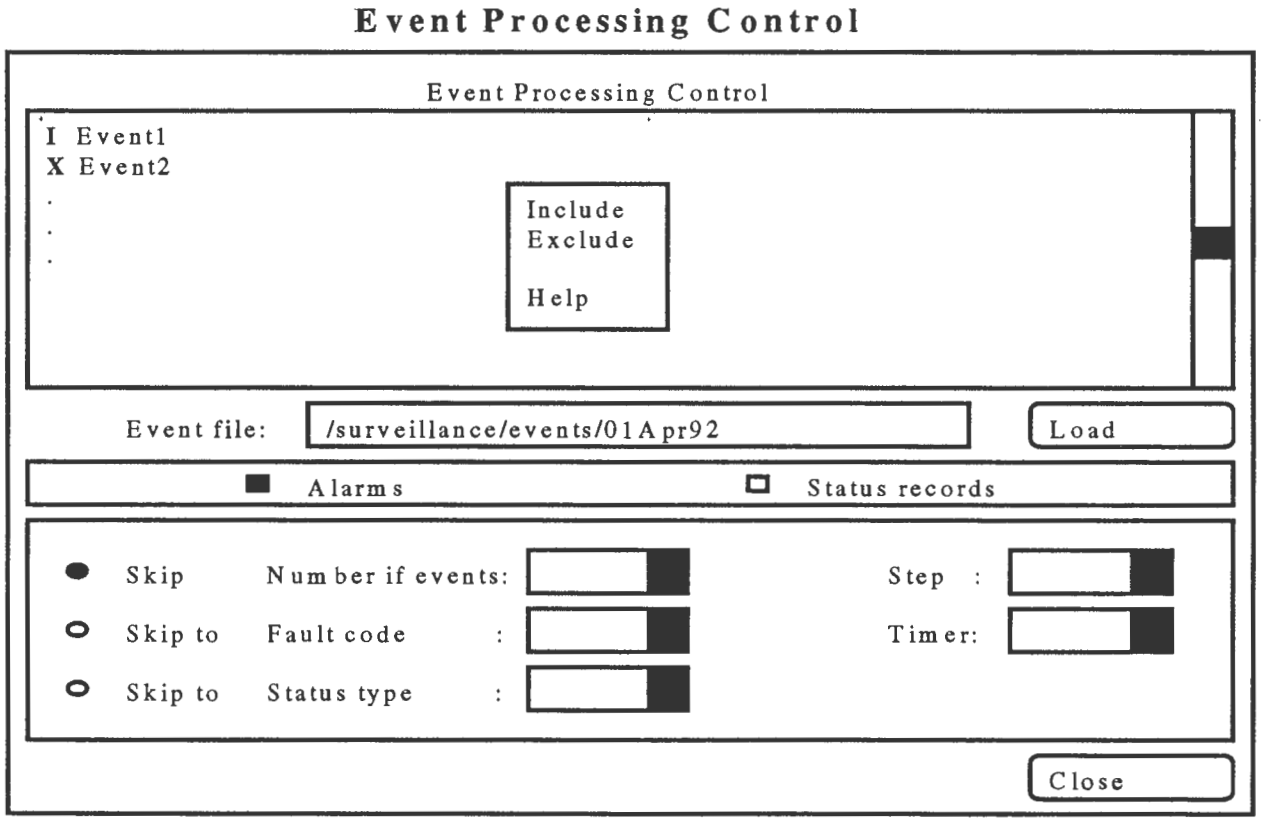


Figure 5.

can be chosen. It is possible to interrupt event processing, step to the next rule or rule action or to resume event processing.

- Specialized browsers are provided in order to display the state of a problem instance, the state variables that are defined for the problem instance, and the values of internal parameters used by the problem instance.

- Watch and break point maintenance. Intelligent cut and paste between debugger browsers is provided in order to fill certain data entry fields automatically. For example, creating a break point on a certain problem instance requires a knowledge of the problem instance identifier, a number. This information is displayed in a browser that displays a summary of all problem instances in the system and thus can be copied from this browser to the browser where break points are defined and maintained.

The main display of the customization tool consists of six distinct areas. The *Script Display Area* is used for display and editing of the source version of a problem script. It is also used to highlight locations of errors in a script during the compilation process. During simulation while stepping through the evaluation of a network event, this area is again used to display the active script and is again highlighted in order to display the current line being evaluated. The *System Message Area* contains a history of the system behaviour and user session progress. For example, this area would contain compilation error messages, diagnostic output from the simulator, knowledge base changes such as loading a new problem script and watch or break point information messages. The *Edit Control Area* consists of two buttons, Edit and Save, and a data entry field named File. This area is used to control the saving of the script displayed in the *Script Display Area* to disk using the revision control system. The *Script Control Area* contains two buttons - Compile and Load. The Compile button invokes the PDL compiler against the displayed problem script. The Load button causes the displayed compiled script to become part of the knowledge base active within the simulator.

The *Simulation Control Area*consists of several buttons that are used to control the progress of a simulation. The Simulation... button causes an Event Processing Control browser to be displayed which allows selection of event file and event filtering. See figure 5 for an example of an Event Processing Control browser. The Start/Suspend button allows a simulation to be interrupted. The Next rule

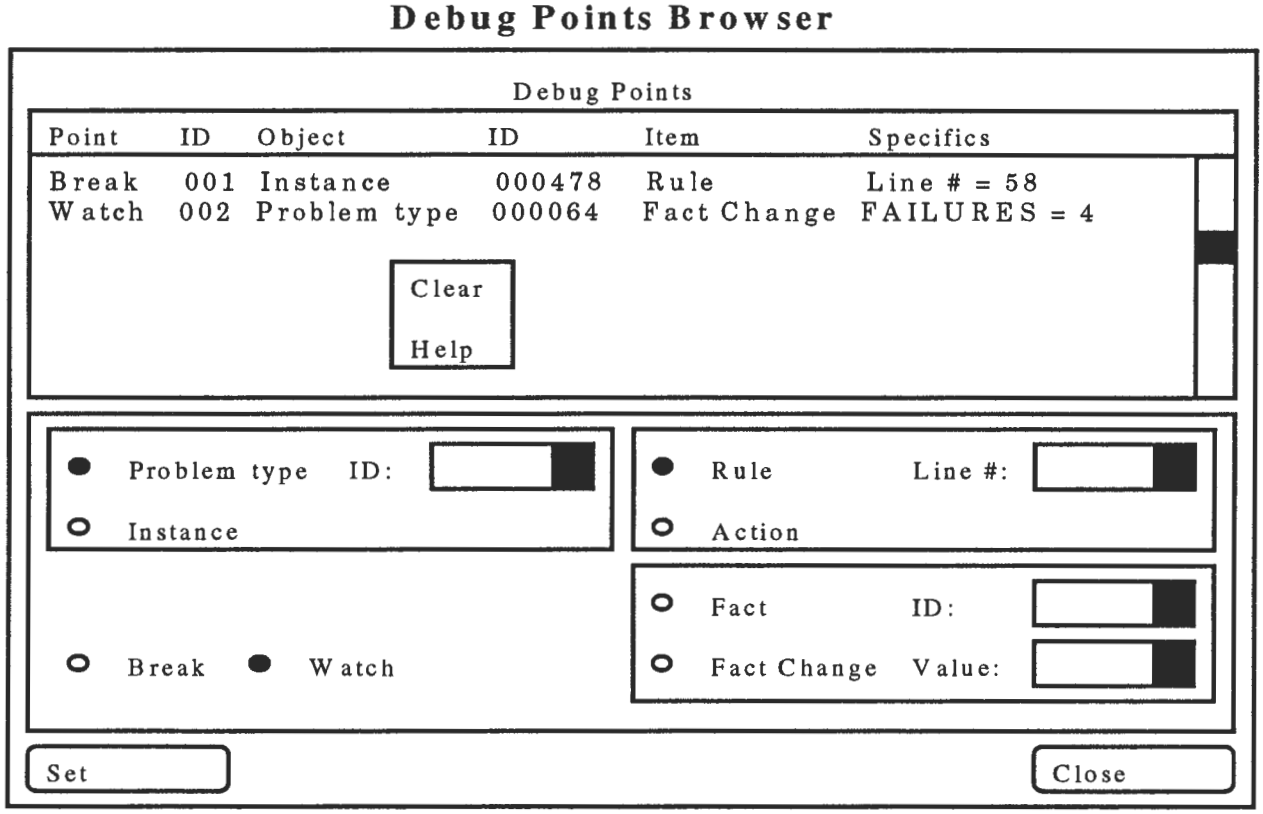


Figure 6.

button allows the simulation to proceed to the next rule, then stop. The Next step allows the simulation to proceed to the next rule or rule action, then stop. The Points... button displays a browser that is used for watch and break point management. The Tracker... button is used to display a browser that contains summary information on the problem instances currently monitored by the simulator. Finally, the *Knowledge Base Area* contains a button that displays a browser containing information on the current state of the knowledge base. This browser gives information on whether a given script has been compiled, where it is loaded and whether it is currently active for the simulation.

Selective Processing of events

The simulator processes network events collected in an event file. The format of the events is the same as generated by the real world network. The stream of events coming from the network can be captured in a file using the real-time expert system or created using a text editor.

Events in the file can be processed selectively, using the features available in the Event Processing Control browser. The user may skip a number of events or discard all events until a specific event is reached. In addition, the user is able to process events in steps with one or more events in each step. The interval between events being processed may be selected in order to provide a slide show capability. Events may be included or excluded (as indicated by **I** or **X** next to the event displayed). When the simulation is suspended, the last processed event is highlighted, so that the progress of the simulation can be observed.

Tracing Problem Script Execution

An event is analyzed using the information encoded in a problem script's rules. A problem script might be executed when a relevant event arrives or a fact changes in value. By following the execution of a problem script we identify the process of evaluation of the rules in the script.

The execution of a problem script can be traced or interrupted by setting debugging break points at rules, actions or on fact changes. Watch points can also be defined which allow reporting to take place whenever a rule, rule action or fact change occurs. When a watch point is reached, a message is generated in the *System Message Area* of the Main Display.

Rule and action debugging points, either watch type or break type, can be set directly from the problem script displayed in the *Script Display Area* of the Main Display with no typing. All types of debug points can be created

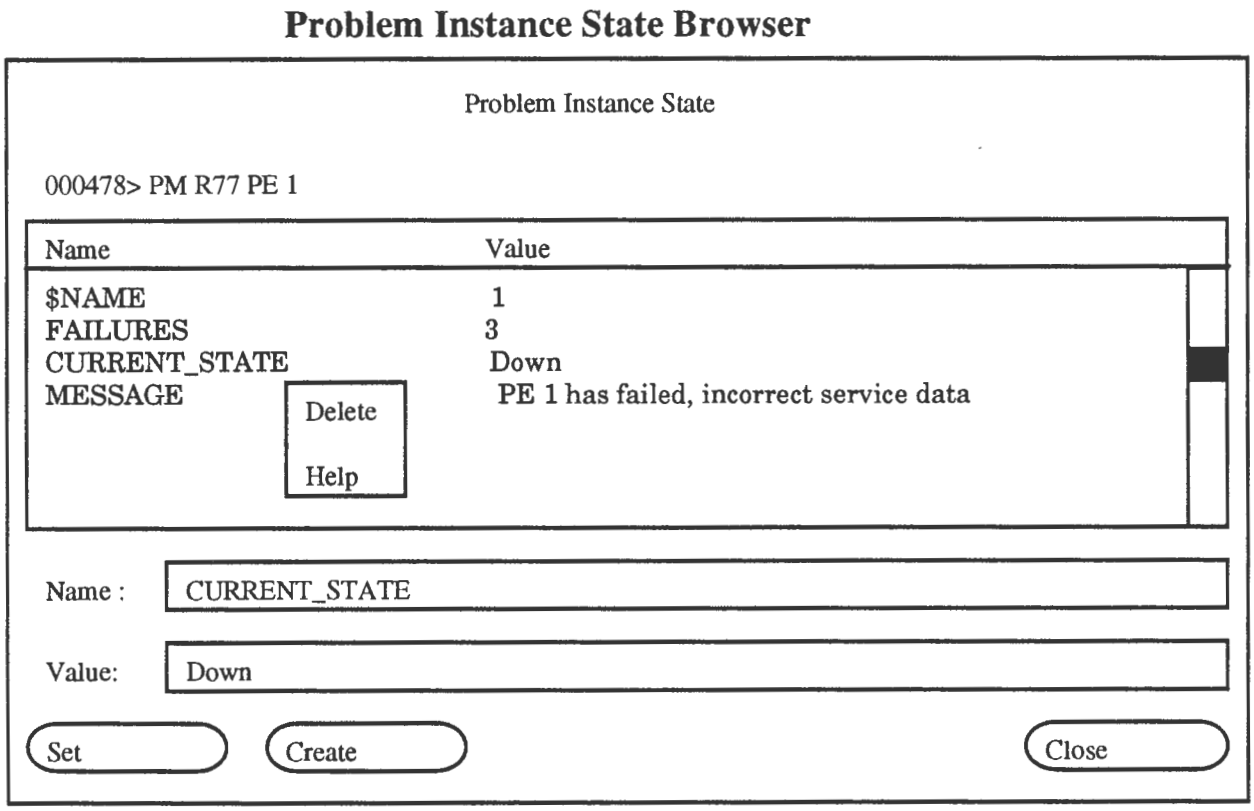


Figure 7.

using the Debug Points browser. Figure 6 shows an example of a Debug Points browser. Intelligent cut and paste of information from other browsers within the customization environment (such as fact identifier, rule line number or rule action line number) is supported in order to reduce the typing required to define a debug point.

When the simulator is suspended at a break point, the *Script Display Area* of the main display of the customization environment contains the source version of the appropriate problem script and the specific break point is highlighted in the Debug Points browser. A message is also added to the *System Message Area* of the main display. The user can control the progress of the simulation using buttons in the *Simulation Control Buttons* region of the main display. It is possible to step through the execution of a problem script, stopping only at the rules being evaluated or also at the rule actions that are evaluated for firing rules.

Reviewing Problem Instance Data

The instantaneous state of a problem instance can be reviewed using a Problem Instance State browser. This browser, shown in figure 7, allows a user to review all facts (or states) currently defined for a problem instance and to modify, or delete them. This browser is used primarily to test the interfaces between problem scripts in the knowledge base, although it can be used to rectify faults in the experimental knowledge base while continuing with a simulation. It may be used to simulate changes in the condition of a problem by changing a fact, which would normally change as a result of a network event.

Summary

In this paper we have described the Expert Advisor Customization Environment, an easy to learn user interface used in the modification and extension of a network surveillance knowledge base. The customization environment was built in order to allow end users of the Expert Advisor to modify their own knowledge base to suit their specific needs. The knowledge base verification mechanism has been described and the value of the modularity in the knowledge base indicated. The environment is used extensively internally both as a training tool and as a way of verifying changes to the knowledge base and underlying software.

By providing a Customization Environment, several benefits have become evident. Firstly, the Customization Environment is a valuable training tool that allows novice network operators to learn at their own pace. This, we believe, will lead to a new generation of network operators with a consistent level of training and knowledge of the network. We have reviewed the PDL with such operators in order to make the language as readable as possible in order that the knowledge base provides a valuable training document in its own right. The what-you-see-is-what-you-get environment allows new operators to become familiar with the problem viewer and its various specialized browsers, in a controlled environment.

The Customization Environment, by allowing the verification of knowledge base changes off-line, ensures that only proven improvements to the knowledge base will be introduced to the operational network surveillance system. Also, allowing on-line changes to the knowledge base to take place, availability of the network surveillance expert system is unaffected.

In conclusion, we believe that this environment provides a significant move forward in operational knowledge based systems. Our current exploratory work with this environment is in the area of visual programming - attempting to remove the dependence on a text-based language.

References

[Kahn, Kepner and Pepper, 1987] Kahn G., Kepner., Pepper J., *TEST: A Model-Driven Application Shell*, Proceedings AAAI 1987 pages 814-818.

[Laffey et. al. 1988] Laffey T., WeitzenKamp S., Read J., Kao S., Schmidt J., *Intelligent Real-Time Monitoring*, Proceedings AAAI 1988 pages 72-76.

[White, Bieszczad, 1992] White T., Bieszczad A., *An Expert System for the Monitoring of Faults in a Packet Switching Network*, submitted to the AAAI Conference, 1992.

Metaknowledge in the LOUTI development system

by Gilbert Paquette
Télé-université, Université du Québec
1001 Sherbrooke est, Montréal H2X 3M4
Telephone: (514) 522-3540
Fax: (514) 522-3608
e-mail: TLRPQU@TELUQ.UQUEBEC.CA

Abstract

This article reports on heuristic guidance in open learning environments. In such environments, when the problem-solving task becomes complex, the number of operations needed to construct interesting results grows exponentially, sometimes discouraging the learner or leading designers and teachers to guide the learner tightly. To prevent the consequent impoverishment of the learner's construction activity, more specific tools must be designed to help him achieve by himself difficult tasks like rule construction, taxonomy definition or quantitative law induction. This last generic task is here analyzed in some detail. First a task model is described, showing the importance of combinatorial metaknowledge. Then, such metaknowledge is represented in a new set of tools for law induction and implemented in the LOUTI development system. Finally, this "generic metaknowledge in tools" paradigm is discussed in the framework of Intelligent Tutoring System research.

1. Introduction

Since 1986, our work has progressed in three main phases. First, commercial generic software - spreadsheets, hypermedia and expert system shells - have been used as development tools to design a first set of learning environments, to experiment them in classrooms and to define a pedagogic strategy [Paquette 88a, Bordier 90].

In a second phase, we have constructed LOUTI, a development system facilitating the design of knowledge-based learning environments [Paquette 88b, Bergeron 90]. Using this workbench, a new set of knowledge-based learning environments has been built. Finally, in the last phase, we have constructed generic problem solving tools and designed counseling agents to guide the learner in his knowledge building activity [Paquette 91].

This paper summarizes this last phase of development in the LOUTI system, focussing on what we call the "generic metaknowledge in tools" paradigm.

1. About Metaknowledge

Many fundamental articles or books have been published on metaknowledge, using various terminologies drawn from domains in cognitive science such as mathematical logic, scientific discovery methods [Popper 61], problem solving and its teaching [Polya 67], computer aided learning [Papert 80], cognitive psychology [Anderson 80] and, of course, artificial intelligence [Minsky 87, Langley 87, Holland 87].

Recently, a broad synthesis has been published [Pitrat 90] drawing from these different domains and viewpoints. Defining metaknowledge as "knowledge about knowledge", Pitrat makes a distinction between passive and active metaknowledge, subdividing the later into five categories: storage and retrieval, communication, evaluation, production, and creation-discovery metaknowledge.

The three last categories play an important role in a constructivist pedagogic strategy. *Evaluation metaknowledge* is involved when a learner is looking at new information and must verify its validity, measure its utility, check if he already has similar information and determine if the information is complete. *Production metaknowledge* is used whenever someone tries to achieve a task or solve a particular problem. Data-driven methods lead to plans where one start with given facts and tries to build new ones until the goal is reached, while Goal-driven methods start from the goal, going backwards to sub-goals until given facts are obtained.

Finally, *Creation and Discovery metaknowledge* is directly involve in any production of new concepts, new rules, new models, or new metaknowledge. For example, *Combinatorial metaknowledge* create new knowledge by applying a set of operators to known knowledge. *Analogical metaknowledge* construct new knowledge by

transforming known knowledge drawn from a similar situation. *Generalization and Specialization metaknowledge* create new knowledge by replacing constants with variables in expressions or restricting/enlarging concept domain definitions or rule conditions.

Metaknowledge can be viewed as a set of cognitive tools for knowledge acquisition in a specific domain. Domain-specific knowledge will be understood and retained to the extent it is evaluated, used or constructed by the learner using some metaknowledge.

On the other end, metaknowledge is a form of knowledge that is more and more valuable in our society invaded by masses of unstructured information. To help learner gain metacognitive skills should be an important goal of computer aided learning systems.

2. General metaknowledge in LOUTI

It has been our main preoccupation to favor metaknowledge acquisition and use in learning environments designed using the LOUTI development system. We will now briefly describe some of the concepts behind LOUTI, identify where metaknowledge is introduced, and underline some limitations in the first version of the system.

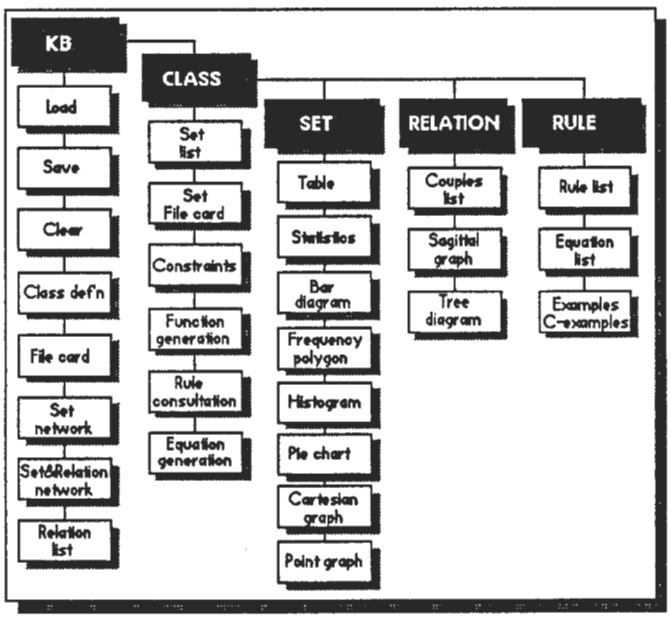


Figure 1

LOUTI is used by a designer to assemble a learning environment. In a first step the designer chooses knowledge representation structures to represent the domain of study - classes, sets, relations or rules, thus selecting a library of possible tools associated with each representation structure. Figure 1 shows structures and their associated libraries, appearing in LOUTI's basic version, completed in April 1990. In the selected libraries, the designer chooses tools that will appear in the learning environment, grouping them in menus. When he is finished, the application thus defined can be assembled automatically. Opening the application, the designer can construct one or more knowledge bases using selected structures and tools. These will be explored by the learner and used to solve problems or achieve tasks. In the process, the learner will often extends the initial knowledge base or constructs an entirely new one.

We will now consider a typical LOUTI application where representation structures are classes, sets and rules. It is intended that the learner analyses classes of experimental observations, define interesting subsets and use rule definition to express relations between certain class attributes. Different knowledge bases can be built and analyzed in this application. Here is a sample interaction with a knowledge base on ideal gas experiment results involving Temperature, Pressure, Volume and Moles.

LEARNER: Reads the task definition: "Open the KB on ideal gases, examine the data using the table tool, and try to find relations between the gases attributes holding for all given observations. If no relation is obvious, try to define a subset of observations where the task will be easier."

LEARNER: Selects the set of all observations, displays a table presentation, use the sorting sub-tool for different attributes and finds not obvious relation.

LEARNER: Using the hint in the task definition, the learner defines a subset of observations where "Temperature = 300 and Moles = 1".

LEARNER: Now looking at the reduced table for this subset, sorting the data for Pressure yield an inverse relation, that can be seen very clearly on the Cartesian graph presentation.

LEARNER: Defines a product attribute Pressure * Volume that seems constant (= 2494.26)

LEARNER: Defines a first rule (law):

"If Temperature = 300 and Moles = 1 Then
Pressure * Volume = 2494.26"

(Using the examples and counter-examples tool gives a clear verification of this law)

LEARNER: Having done the same for other constant values of Temperature and Moles, defines a more general subset of observation where "Temperature = 300".

LEARNER: Now looking at the two attributes Pressure*Volume and Moles, sorting the data for Moles yield a direct relation, that can also be seen on a Cartesian graph presentation of these two attributes.

LEARNER: Defines a quotient attribute (Pressure*Volume)/Moles that is constant (= 2494.26)

LEARNER: Defines another rule (law):

"If Temperature = 300 Then
(Pressure*Volume)/Moles = 2494.26"

LEARNER: Selects the initial set of observations,

displays a table presentation of the last defined attribute and Temperature; sorting on Temperature yields another direct relation.

LEARNER: Defines a rule (law) holding for all the observations given initially in the KB:

"If t Then (Pressure*Volume)/(Moles*Temperature) = 8.3142

The striking thing in the preceding example is that it is a monologue, not a dialogue. This is characteristic of open learning environments. The learner directs the action, sometimes aided by a human teacher who suggests methods and tools, getting ideas for future action from the system's reaction. The system's role is to table, to sort, to graph, to accept set and rule definition, to adapt its presentation tools to newly selected set or rule, to give a clear presentation of a rule's examples and counter-examples for verification by the learner.

This is possible in the basic version of LOUTI, because tools embody general metaknowledge about knowledge processing in the same way a text editor embodies general metaknowledge about text processing. A text editor knows that reordering sentences or paragraphs, searching and replacing words, indenting paragraphs or changing character style, are important to good communication in any particular knowledge domain. In a similar way, a LOUTI application adjusts its subsets inclusion network after set definition (figure 2A), adjusts tables and graphs to set selection (figure 2B), facilitate sorting operations and rule counter-examples display (figure 2C).

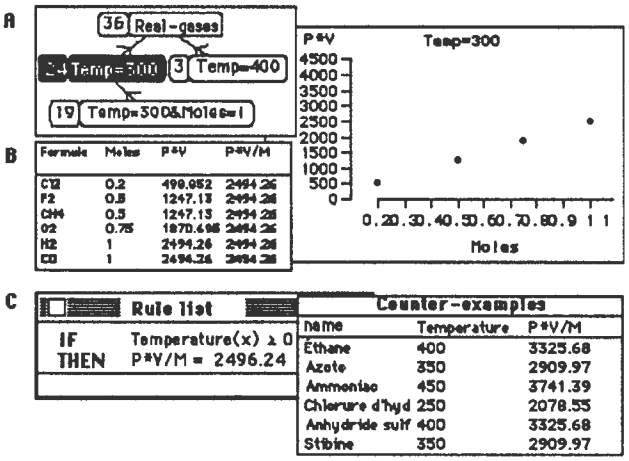


Figure 2

There are however, problems in using tools that are as general as these. It is not certain at all that the average student will be able to achieve anything near the ideal method presented above. In fact, in the first implementation of this learning environment, the designer has been forced to write a detailed activity guide to make sure that the learner proceeds in the right direction.

Decomposing the learning process in a "guided tour" way is contradictory to the very idea of a constructivist learning strategy. It leads to impoverishment of the problem-solving process, preventing the learner to plan the use of induction tools and methods by himself

The problem here is this: in a complex learning task, the number of necessary operations using general tools grows exponentially. To prevent this without guiding closely the learner, the LOUTI workbench must be enriched with new tools that are concrete representation of more specific metaknowledge.

A first set of such tools have been developed with the SONODOSE project, a physiotherapy training environment on ultra-sound treatment [Paquette 90]. Specialized tools have been constructed to help the learner build a consistent, complete and non-redundant set of rule for ultra-sound treatment identification. Evaluation metaknowledge has been embedded in tools to help uncover contradictions between rules, compare the extension of two rules and assess a rule-group's completeness. These tools are not as general as LOUTI's base version tools, but they are useful whenever the generic task of rule construction must be undertaken.

To generalize this approach, we have since then developed new set of tools for two other generic tasks: taxonomy definition and quantitative law induction. The later will now be discussed in some detail, keeping in mind to help a learner solve similar induction problems to the Ideal Gas law.

3. Law induction metaknowledge

In law induction tasks, fundamental AI machine learning research [Langley 87] has demonstrated the importance of combinatorial metaknowledge. In the work of Langley and al. on scientific discovery, induction of quantitative laws is seen as a problem-solving process in a state-operator space. The initial state is a list of expressions that are variables over some domain. Operators can restrict, enlarge the domain or construct more complex expressions. They are applied until a final state is reached where an expression is a constant or a linear expression over a sufficiently large domain, or no more operators can be applied.

The search through the induction problem space is guided by combinatorial metaknowledge. Based on our observation of human problem solvers, using part of Langley's BACON programs heuristics, we have retained the following operators or metaknowledge:

TO INDUCE:

(1) Restrict the domain so that all variables are kept constant, except two called X and Y, and make a first expression list X,Y on this domain.

(2) If X and Y are inversely proportional, add the expression X*Y to the list.

(3) If X and Y are directly proportional, add the expression X/Y to the list.

(4) If the last expression F(X,Y) is constant

or a linear function of X and Y, remove X and Y from the expression list.

(5) If the last expression F(X,Y) is constant or a linear function of X and Y, stop or enlarge the domain by freeing one variable not in F(X,Y).

Using these operators, the direct induction path to the Ideal Gas law presented in the preceding section, can be summarized as follows:

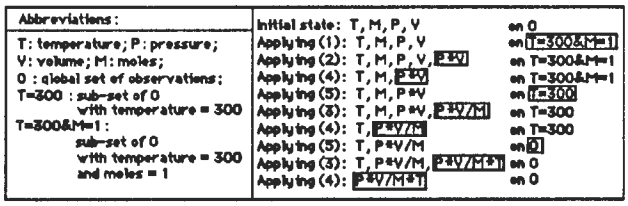


Figure 3

Metaknowledge in these operators is generic. With similar heuristics, the BACON programs are able to discover most quantitative laws of elementary physic and chemistry such as Snell's refraction law, Ohm's electricity law or Kepler's third gravitational law. Furthermore, these heuristics were derived from close study of the writings of scientists involved in these laws' first discovery. So it seems a sound pedagogical idea to include them in some way in a new set of LOUTI tools.

4. Law induction metaknowledge in tools

Our implementation of the above operators has led to four new LOUTI tools:

- **Variable Selection** displays, for any selected set, a list of attribute couples, each being in inverse or direct proportional relation, to help the learner apply operators (1) & (5). (see Figure 4 A, B)
- **Construction** displays a sorted table of the selected set, with a column for the two initial variables and subsequent expressions constructed using operators; as shown on figure 4C, operators (2) Product, (3) Quotient and (4) Reduce can be applied to the current expression list using options of the private menu.
- **Operation Selection** counsels the learner on the operation to use on two attributes he selects.
- **Constant and Linear Testing** computes if an expression is a constant or linear relation, considering an error factor specified by the learner.

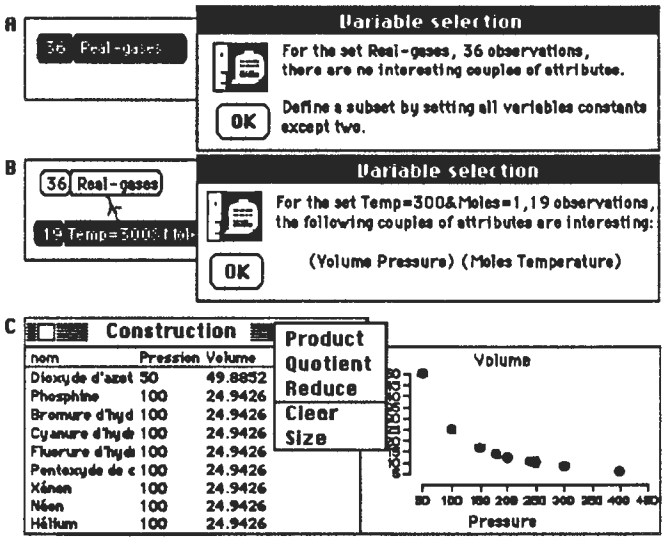


Figure 4

Using these new tools, the Ideal Gas law induction is more easily achieved, the learner proceeding as follows:

LEARNER: Reads the task definition: "Open the KB on ideal gases, examine the data using the table tool, and try to find relations between the gases attributes holding for all given gases".

LEARNER: Calls the Variable Selection tool with the set of all observations selected.

SYSTEM: Suggest the definition of a restricted subset (Figure 4A).

LEARNER: Defines the Temp=300&Moles=1 subset and calls again the Variable Selection tool.

SYSTEM: Suggest a relation between the variables Pressure and Volume (Figure 4B).

LEARNER: Calls the Construction tool; the sorted table and the associate Cartesian graph clearly show an inverse proportional relationship (Figure 4C).

LEARNER: Calls the Operation Selection tool

SYSTEM: Suggest to consider a product of the two variables.

LEARNER: Selects the product option in the Expression Construction tool; the table and graphics adjust, showing a constant product expression (= 2494.26), verified to a very low precision interval using the Constant Testing tool.

LEARNER: Uses the "Reduce" option of the Construction tool, keeping expression T, M and P*V

LEARNER: Defines a more general subset of observation where "Temperature = 300" and call the Variable Selection tool.

SYSTEM: Suggest to consider a relation between the variables Moles and P*V.

LEARNER: Calls the Construction tool; the sorted table for the subset and the associate Cartesian graph clearly show a direct proportional relationship between Moles and P*V.

LEARNER: Selects the quotient option in the Construction tool; the table and graphic adjust, showing a constant P*V/M expression (= 2494.26) that can be verified using the Constant testing tool.
LEARNER: Uses the "Reduce" option in the Construction tool, keeping expression T and P*V/M for future constructions.
LEARNER: Select the initial "Real Gases" observation set.
LEARNER: Calls the Construction tool; the sorted table and the associate Cartesian graph clearly show a direct proportional relationship between T and P*V/M.
LEARNER: Selects the quotient option in the Construction tool; the table and graphic adjust, showing a constant P*V/M*T expression (= 8.3142), yielding the Ideal Gas Law definition.
LEARNER: Defines a rule (law) holding for all the observations given initially in the KB:
"If t Then (Pressure*Volume)/(Moles*Temperature) = 8.3142

Comparing this procedure and the one in section 2, one can see that the new set of tool reflects more directly induction heuristics. The learner is freed from most technical operations so he can concentrate on finding a solution path. Further more, such tools are constant reminders of productive induction methods.

Though the learner retains initiative, the system is more active, giving advice at the student request on interesting variable, adequate operations and constancy or linear relationships. While using the tools, the learner has to ask himself good questions about the method.

The end of the preceding dialog suggests that the learner has come to associate "inverse proportional relation" with a product operation, and "direct proportional relation" with a quotient operation, so doesn't need to use the Operation Selection tool. When the learner comes to find some tool useless, and yet use its effect to choose other actions, we can suppose he has internalize the corresponding metaknowledge. Then, a very important didactic goal has been achieved.

5. Implementing Generic Tools in the LOUTI system

In the LOUTI system, tools are PRISME[1] objects inheriting attributes and methods from parent tools or basic interface objects. To add a new tool, it is useful to start with one or more general tools already in the system and to modify their methods.

[1] PRISME [Bergeron 88, Nadeau 91] is a programming system of the LISP family with object-oriented and logic programming extensions, fully integrated with one another and with the Macintosh graphic user interface.

Operational tools

We define tools as operational, when they participate in a learner's constructive action. They facilitate action, in our case, applying operators in the problem-space.

For example, the *Construction tool* is implemented as a descendant of both the Table Tool and the Cartesian Graph Tool, but almost every method had to be rewritten. Activation and deactivation methods needed to be redefine, to achieve coordinate appearance or disappearance of the sorted table and its associated graphs.

The tool's private menu has been profoundly modified to make the Product, Quotient and Reduce options available. These three options achieve several knowledge base operations in a single step:
- call attribute definition or destruction operations,
- specificy attribute type and computation mode,
- construct new product or quotient attribute,
- add a column in the sorted table and draw new graphs.

Counseling tools

Unlike operational tools, counseling tools give methodological advice to the learner. They are implemented as descendants of a parent tool having two basic methods:

- evaluating a set of conditions based on indicators to be computed by each counseling tool;
- displaying a message based on the conditions met by the learner's knowledge base in its present state.

A particular counseling tool like *Variable Selection* will compute indicators for direct or inverse proportional relationship for each pair of attributes, for all elements in the selected set of observations. It will display a message accordingly. If there is no such relationship, it will suggest the learner to apply operator (1). If, on the contrary there is one, it will list couples of interesting attributes to consider.

Such counseling tools do not take action on behalf of the learner. Like any other tool in the environment, a counseling tool is used by the learner whenever he feels he needs it. Getting an advice leaves a lot of work to do. There are still many paths the learner can take. If fact, counseling tools act like a first-line specialized teaching agent. The learner mentally raises his hand and gets a specialized counsel from the agent.

Generic tools

Both operational and counseling tools are domain independent. This opens the possibility to add them to LOUTI's libraries. They then become available to build any application in which they can be useful. These are knowledge domains where there is at least a class containing a certain number of numeric attributes on which subsets, can be defined and numerical relations searched for.

The law induction tools are domain independent but they display messages, tables, graphs or other presentations using domain specific terms. For example, figure 5 shows displays

from the Variable Selection tool and the Construction tool, in a completely different domain from the Ideal Gas law: solar astronomy. Thirty-nine celestial objects are includes in the knowledge base and described by more than thirty attributes, most of which are numeric. It is then possible to use the Law induction tools to induce, for example, Kepler's third gravitational law.

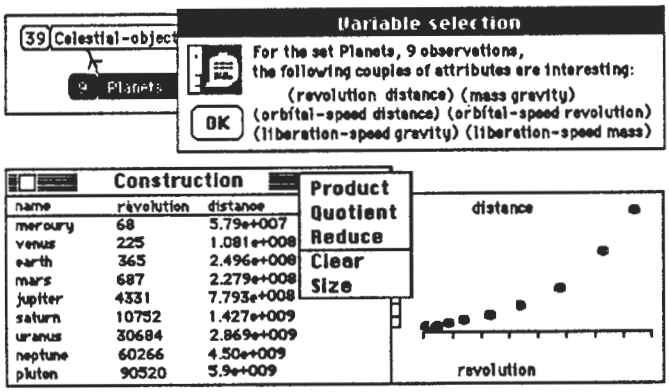


Figure 5

This induction problem-space is quite different from the ideal gas law problem. Kepler's third law contains only two variables instead of four, but the relation between them is more complex. To construct the expression D3/R2 = cst, one does not need to restrict or enlarge the observation set, but to apply the product and quotient operators many times on expressions obtained by previous application of the same operators.

The fact that a law such as this one can also be derived using the same induction tools shows their generic character. But certainly they are not general tools, useful in any induction problem. Adding new operators would enable induction of more complex quantitative laws, but for ill-structured problem domains such as quasi-laws in economics or foreign policy, combinatorial metaknowledge such as the one presented here would prove its limit. For such domains, rule construction metaknowledge such as we have embedded in the physiotherapy environment might provide a better start.

This discussion underlines our general approach: to find classes of similar problems where useful metaknowledge can be identify, to embed this metaknowledge both in operational tools and counseling tools, and to make these tools available to the learner in the learning environment.

6. Learner Guidance in Open Learning Environments

We have presented an approach to the problem of guidance in open learning environments. Especially when problems are difficult, it is necessary for the system to provide some sort of advice, without restraining the necessary liberty involved in real problem-solving activities. We will now discuss this approach more in detail, within the framework of Intelligent Tutoring Systems (ITS) research.

The learner as a knowledge engineer

The following quotation from John Self expresses well the foundation of our approach:

> "An ITS philosophy (of Knowledge transmission) runs counter to almost everything of significance in twentieth century educational philosophy. All the major figures (Dewey, Montessori, Piaget, Rodgers, even Skinner) have rejected the "education as transmission" model, which had dominated the previous three centuries in favor of an "education as growth" model. Knowledge is not the kind of commodity which can be transmitted. It cannot be simply received by students but must be constructed anew by them. While most ITS researchers declare such a view, it is belied by the systems we build." [Self 90]

We hope we are not! In fact, in our learning environments, the student's activity is essentially knowledge construction. The learner's explores the knowledge freely using presentation tools in the application. Then, a task or a problem makes him explore knowledge in a more structured way. Finally he will write down a conjecture, test it and modify it if necessary to increase its predictive power.

Many works in ITS have considered the analogy between the learner's construction activity and knowledge engineering. For example, William Clancey proposes the following approach:

> "by studying and modeling the work of the knowledge engineer, we should be able to build models of the learning process that can be integrated in the design of an Intelligent Tutoring System." [Clancey 88]

While Clancey here looks at the knowledge engineer as a learner to model, we essentially see the learner as a knowledge engineer. In both cases, the study of knowledge engineering is interesting because it helps identify more clearly useful metaknowledge. Since knowledge engineering by the student is not and easy task we seek to make metaknowledge available in the form of interactive tools included in the learning environment.

Metaknowledge in tools and heuristic guidance

We are certainly not the first to investigate the use of metaknowledge in learning environments. Clancey has used metaknowledge classified as support, structural and strategic knowledge and used in his systems for medical training [Clancey 83]. Vivet has integrated the mathematician metaknowledge in an expert-system solving symbolic integration problems, that can be used for tutoring a student on those tasks [Vivet 84].

In open environments, Papert's microworld approach [Papert 80] puts great emphasis on the use of problem-solving metaknowledge through LOGO programming.

Learning environments such as SMITHTOWN [Shute 86], REFRACT [Reimann 88] and VOLTAVILLE [Glaser 88], built at Pittsburgh's LRDC have a strong metacognition emphasis.

In our view, metaknowledge should be introduced explicitly in learning environments to help the student gain and use domain independent methodological knowledge. This idea is supported by a systematic study of the use of LOGO problem-solving in classrooms [Swan 89]. This study shows that LOGO programming, though having an effect on problem-solving skills, is not significantly better without explicit teaching of problem-solving methods such as problem decomposition, forward-chaining or analogy. Both are necessary for metaknowledge use and acquisition.

In a similar way, we hypothesize that the use by the learner of operational tools embedding metaknowledge, and the use of counseling tools, also based on metaknowledge, are both necessary to help the student achieve knowledge construction tasks.

Operational tools enable activity similar to LOGO program construction, with its programming-evaluating-debugging cycle. Using operators on the knowledge base, the learner construct a problem solution by a similar "knowledge debugging" process.

On the other hand, counseling tools are a constant reminder of evaluation, utilization and creation metaknowledge, in other words: problem-solving methods. They provide an interactive initiation to problem-solving methods, a form of heuristic teaching first proposed by George Polya [Polya 57, 67]. In this way, in a very open learning environment and task, it is possible to offer methodological guidance that should constrain the leranrer's construction process towards productive results.

Generic metaknowledge

A last aspect of the question is the generality or scope of metaknowledge. In their synthesis on induction, Holland and al. underline:

> "General methods such as means-ends analysis are not sufficient to explain problem-solving skills. Human expertise lies, in a critical way, on specialized methods and a good representation of knowledge in the considered domain" [Holland 87]

In their work on scientific discovery, Langley and al., also consider a hierarchy of heuristics (or metaknowledge) from general weak methods, to more specialized strong methods:

> "We can picture a hierarchy of heuristics, with the top levels consisting of very general algorithms that require little information about the task domain to which they are applied and are correspondingly applicable to a great many domains. As we proceed downward in the hierarchy of heuristics, we make more and more demands on information about the task domain, and there are correspondingly fewer domains to which the heuristics apply (...). Since the more general heuristics operate on the basis of less information, we may expect them to be less selective, and hence less powerful, than heuristics that make more use of information about the task structure. Hence, general heuristics are weak methods, while task-specific heuristics, such as our procedure for solving algebra equations, may be strong methods. The less structured a problem-solving task is, the less information one can draw upon construct strong heuristics." [Langley 87]

Reviewing recent ITS literature shows that the vast majority of projects have concentrated on strong, domain-specific methods, consistent with a guided discovery approach. Metaknowledge has been used mainly in the tutorial module to achieve close domain-dependant learner's guidance. At the opposite, micro-world and learning environment projects are characterized by the use of weak methods, most of the time without any guidance from the system.

Our work is an attempt to reconcile these approaches. We have first represented weak methods in quite general tools, in the basic version of the LOUTI system, in a typical "micro-world spirit". Then, we have concentrated on the introduction of more specific metaknowledge and stronger methods for generic tasks like rule construction, and law induction.

We can define our research program as this: to seek structured tasks where powerful, while largely domain-independent methods can be represented in tools to support a learner-centered construction activity.

It is hoped that with these tools, a learner can experiment fruitful knowledge construction activities across different fields of knowledge.

Acknowledgements

The author wishes to underline the contribution of Anne Bergeron, Jacques Bordier, Charles Camirand, Serge Carrier, Annick Hernandez, Martin Longpré and Renaud Nadeau to the LOUTI project, and the support of the APO-Québec Center and Télé-université.

References

[Anderson 80] J. Anderson. *Cognitive Psychology and its implications.* Freeman and Co., 1980

[Bergeron 88] Bergeron A., Bouchard L., Nadeau R., *A Multi-Paradigm Development System for Exploratory Environments,* CSCSI 88 Proceedings, Edmonton,1988.

[Bergeron 90] Bergeron, A. and Paquette, G. *Discovery Environments and Intelligent Learning Tools,* in Intelligent tutoring systems: at the crossroad of Artificial Intelligence and Education (Eds C. Frasson and G. Gauthier), Ablex Publishing, Norwood NJ.

[Bordier 90] J. Bordier, G. Paquette and S. Carrier, *Building Learning Environment using Generic Software*, Proceedings of the 4th World Conference on Computers and Education, Sydney, Australia, July 1990.

[Clancey 83] W. J. Clancey. *The Epistemology of a Rule-Based Expert Systems.* Artificial Intelligence 20, pp 163-210, 1983.

[Clancey 88] W. J. Clancey. *The Knowledge Engineer as Student: Metacognitive Bases for Asking Good Questions,* in Learning Issues for Intelligent Tutoring Systems, H. Mandl & A. Lesgold (Eds), Springer-Verlag, 1988.

[Glaser 88] R. Glaser, K. Raghavan and L. Shauble.*Voltaville, a Discovery Environment to Explore the Laws of DC Circuits.* Proceedings of the ITS-88 conference, Montreal, June 1988.

[Holland 87] J.H. Holland, K.J. Holyoak, R.E. Nisbett, P.R. Thagard. *Induction, Processes of Inference, Learning, and Discovery*, MIT Press, Cambridge, 1987.

[Langley 87] P. Langley, H.A Simon, G.L. Bradshaw, J.M. Zytkow, *Scientific Discovery, Computational Explorations of the Creative Processes,* MIT

[Minsky 88] M. Minsky. *Society of Mind* , MIT PRESS, 1987.

[Nadeau 91] R. Nadeau. *Intégration de trois paradigmes de programmation,* Mémoire de maîtrise en mathématique and informatique, UQAM, 1991.

[Papert 80] S. Papert. *Mindstorm: Children, Computers and Powerful Ideas,* Basic Books, 1980.

[Paquette 88a] G. Paquette. *Environnements d'apprentissage à base de connaissances.* Conférence invitée au colloque international de pédagogie informatique, Actes publiés par l'Université de Bologne, Italie, juin 88.

[Paquette 88b] G. Paquette. *Le développement d'outils intelligents d'apprentissage pour le traitement des connaissances.* Actes de ITS-88, U. de Montréal, June 1988.

[Paquette 90] G. Paquette, R. Nadeau and M. Longpré. *A support-system for Knowledge Engineering by the Student*, CSCSI-90, Ottawa, may 1990.

[Paquette 91] G. Paquette, *Le rôle des métaconnaissances dans LOUTI, un générateur d'environnement d'apprentissage,* Actes du colloque KMET'90, IOS Press, Amsterdam, avril 1991.

[Pitrat 90] J. Pitrat. *Métaconnaissance, avenir de l'intelligence artificielle*, Hermès, Paris, 1990.

[Polya 57] George Polya. *How to Solve It?*, Princeton University Press, 1957.

[Polya 67] George Polya. *La découverte des mathématiques,* Wiley, Dunod Paris, 1967.

[Popper 61] K.R. Popper, *The logic of scientific discovery,* Science editions, 1961.

[Reimann 88] P. Reimann, *Modeling discovery learning processes in a microworld for optics,* European summer university on intelligent tutoring systems, Université du Maine, Le Mans, 1988.

[Self 90 J. Self. *Theorietical Foundations of Intelligent Tutoring Systems* Journal of Artificial Intelligence in Education, AACE, Phoenix Arizona, vol 1-4, summer 1990.

[Shute 86] V.J. Shute and J.G. Bonar. *An intelligent tutoring system for scientific inquiry skills.* Proceedings of the Eighth Cognitive Science Society Conference, Amherst , Massachusetts, 1986.

[Swan 89] Karen Swan, *Logo Programming and the Teaching and Learning of Problem Solving,* Journal of Artificial Intelligence in Education, AACE, Phoenix Arizona, fall 1989.

[Vivet 84] M.Vivet. *Expertise mathématique and informatique: CAMELIA, un logiciel pour raisonner and calculer.* Thèse d'état, Université de Paris VI, 1984.

[Wenger 87] Etienne Wenger, *Artificial Intelligence and Tutoring Systems, Computational and Cognitive Approaches to the Communication of Knowledge,* Morgan Kaufman, Los Altos, USA, 1987.

Computing The Lower Bounded Composite Hypothesis By Belief Updating

Yang Xiang
Expert Systems Laboratory
Centre for Systems Science
Simon Fraser University
Burnaby, B.C., Canada, V5A 1S6
yangx@cs.sfu.ca

Abstract

When making a decision under uncertainty, we often impose a threshold on the expected utility. Such a threshold dictates a lower bound for the posterior probablity of the composite hypothesis on which the decision is based. This paper introduces the concept of Lower Bounded Composite Hypothesis (LBCH). A LBCH is also a Most Likely Composite Hypothesis (MLCH) if the lower bound is greater than 0.5.

The paper presents a threshold method for computing LBCHs by belief updating (the computation of Posterior Marginal Distributions (PMDs)) in Bayesian networks. There has been no known algorithm computing MLCHs in clique tree based secondary structures of Bayesian networks. The method presented allows the computation of LBCHs in such structures using existing belief updating algorithms. The method is practical only for computing LBCHs over a small number of (about 20) variables. However, the networks can contain an arbitrarily large number of variables. It is argued that restricting the computation of LBCHs to a small number of variables can often be desirable. An example is given to illustrate the method.

1 Introduction

Bayesian belief networks combine graphic representation of causal domain models and probability theory. Over the last decade, they have gained increasing popularity as a natural, efficient knowledge representation method and a consistent inference formalism for building knowledge based systems which require reasoning under uncertainty.

Bayesian networks have been termed in the literature *belief networks, causal networks*, or *causal probabilistic networks*. Formally a Bayesian network [Pearl 88] is a triplet (N, E, P).

- The *domain* N is a set of nodes each of which is labeled with a random variable. Each variable has a set of mutually exclusive and exhaustive outcomes which forms its space. 'Node' and 'variable' are used interchangeably in the context of Bayesian nets. A joint assignment of outcomes for a set of variables $X = \{A_1, \ldots, A_n\} \subseteq N$ is denoted by the concatenation of corresponding outcomes $x = a_1 \ldots a_n$.
- E is a set of arcs such that (N, E) is a directed acyclic graph. The arcs signify the existence of direct causal influences between the linked variables. Causality is directed along the directed arcs. The basic dependency assumption embedded in Bayesian nets is that a variable is independent of its non-descendants given its parents.
- P is a joint probability distribution quantifying the strengths of the causal influences signified by the arcs. P specifies for each $A_i \in N$ the distribution of the random variable labeled at A_i conditioned by the values of A_i's parents π_i in the form of a conditional probability table $p(A_i|\pi_i)$. $p(A_i|\pi_i)$ is a normalized function mapping the joint space of $\{A_i\} \cup \pi_i$ to $[0, 1]$. The joint probability distribution P can be represented as $P = p(A_1, \ldots, A_\alpha) = \prod_{i=1}^{\alpha} p(A_i|\pi_i)$.

Inference in Bayesian networks can provide answers to essentially two types of queries. The first type asks the PMDs in the form: $p(A|\mathbf{e})$ where A is a *single* variable and $\mathbf{e}$ stands for all the evidence acquired so far. Computing PMDs is termed *belief updating* [Pearl 88]. A PMD allows one to rank the set of possible outcomes for variable A according to their credibility given $\mathbf{e}$, for example, whether a patient has a particular disease or not; or whether a component in a machine is faulty.

The second type asks the MLCH. Let $X \subseteq N$ be a set of variables. One computes a joint assignment x of outcomes to all variables in X such that

$$(\forall x')p(x|\mathbf{e}) > p(x'|\mathbf{e}).$$

A joint assignment x' is called an *instantiation* of X, or a *composite hypothesis*, or a *configuration*, or an *explanation*. Computing the MLCH is termed *belief revision* [Pearl 88]. The MLCH is needed when one tries to determine the exact state of the world (when $X = N$) [Poole and Provan 90], for example, to give a description of the overall status of a patient.

In the literature, the two types of inference have been considered distinct. Consider two variables A and B.

The computation for PMDs may indicate that a_1 is the most likely outcome of A and b_1 is the most likely outcome of B. While the MLCH for $X = \{A, B\}$ may turn out to be $a_1 b_2$ [Pearl 88, Poole and Provan 90].
This paper shows when this is not the case. Based on the distinct nature of the two inference types, separate algorithms have been developed for each type of inference. For computation of *exact* PMDs, there are $\lambda - \pi$ message passing in trees [Pearl 86], arc-reversal and node-removal [Shachter 86, Shachter 88], directed clique tree message passing [Lauritzen and Spiegelhalter 88], junction trees of cliques [Jensen, Lauritzen and Olesen 90a, Jensen, Olesen and Andersen 90b], multiply sectioned Bayesian networks and junction forests of cliques [Xiang, Poole and Beddoes 91]. For computation of *approximate* PMDs, there are logic sampling [Henrion 88], and clamped logical sampling [Pearl 88]. For computation of the MLCH, there are message passing, cutset conditioning, clustering [Pearl 88] as *exact* methods; and TopN [Henrion 91] as an *approximate* method.

This paper studies the relation between the two types of inference. In particular, the paper presents the following. (1) The notion of *Lower bounded Composite Hypothesis* (LBCH) is introduced from a decision theoretic context. LBCHs are MLCHs when the lower bound is greater than 0.5. (2) It is shown that LBCHs can be obtained by belief updating. (3) Although computation of LBCHs by belief updating is directly applicable to hypothesis sets of less than about 20 variables, it is argued that not only this is often sufficient with respect to one's current attention, but this is often desirable compared to computing a MLCH over all domain variables. (4) There is no known algorithm computing MLCHs in clique tree based secondary structures of Bayesian networks. By establishing the relation between belief updating and revision, one can extract extra information (LBCHs) beyond PMDs from the secondary structures in a straightforward way.

Section 2 introduces the LBCH notion from maximum expected utility principle of decision theory. Section 3 and 4 presents the threshold method for computing LBCHs by belief updating. Section 6 discusses the application of the method for inference in secondary structures. Section 7 demonstrates the method by an example. Section 8 summarizes the main results of the paper.

2 Lower Bounded Composite Hypotheses

Decision theory argues a rational decision maker should act according to Maximum Expected Utility (MEU) principle [von Neumann and Morgenstern 47]. Following the formulation in [Andreassen et al. 89], the decision maker's preference can be represented by a utility function $U(x_i|x_j)$ where x_i and x_j are both composite hypothesis on a set X of variables. The function specifies the utility of action based on x_i given that x_j is true. We consider reward/penalty type of utility functions. The utility of a correct decision ($i = j$) is positive (reward). The utility of at least one wrong decision ($i \neq j$) is negative (penalty). Utility functions in many applications belong to this type. Military actions on battlefields are such examples. In medical diagnoses, treatments (medication, surgery) based on wrong diagnoses can cause side effects and inappropriate follow-ups, can prolong and worsen the patients' suffering due to the untreated real disease. There has been argument [Huchlenbroich 91] that medical expert systems should base their conclusion on "conclusive" evidence to avoid incorrect diagnosis as much as possible.

The MEU principle says that given evidence e, and utility function U, one should choose the action indicated by x_i which has maximal expected utility:

$$\overline{U}(x_i) = \max_k (\sum_j U(x_k|x_j) p(x_j|\mathbf{e}))$$

Denote the reward/penalty type utility as the following.

$$U(x_i|x_j) = \begin{cases} V & i = j \\ w_j & i \neq j \end{cases}$$

where $\min(w_j) = -W$, and V, W are positive. The expected utility for x_i is

$$\begin{aligned} \overline{U}(x_i) &= V p(x_i|\mathbf{e}) + \sum_{j \neq i} w_j p(x_j|\mathbf{e}) \\ &\geq V p(x_i|\mathbf{e}) - W(1 - p(x_i|\mathbf{e})) \\ &= W(1+r)(p(x_i|\mathbf{e}) - p') \end{aligned} \quad (1)$$

where $r = V/W$ (*reward/penalty ratio*) and $p' = 1/(1+r)$.

When making a decision under uncertainty, we often impose a threshold U_0 on the expected utility. If MEU is less than U_0, we would delay the decision and try to gather more information until MEU is greater than U_0. The following definition gives one such threshold.

Definition 1 *With a reward/penalty type utility, a composite hypothesis x is* **profitable** *if and only if $\overline{U}(x) > 0$.*

By equation 1, if $p(x_i|\mathbf{e}) > p'$, then x_i is profitable. Consider an extreme case of the reward/penalty ratio in equation 1. If $V << W$, then $p' \rightarrow 1$. A profitable composite hypothesis x_i has to satisfy $p(x_i|\mathbf{e}) \rightarrow 1$. In this case, trying to find the profitable composite hypothesis means trying to avoid heavy penalty. For instance, when a medication can cure a patient if he has the disease but can kill him if the diagnosis is wrong, the physician should be very careful to ascertain the diagnosis. We see that p' is a belief threshold which results from the expected utility threshold 0.

Zero is not the only possible utility threshold. Consider another extreme case of the reward/penalty ratio in equation 1. if $V >> W$, then $p' \rightarrow 0$ and the right side of the equation is approximately $V p(x_i|\mathbf{e})$. In such case, many composite hypotheses are profitable but the reward associated may be very low compared to V. If one sets minimum expected reward V_0, then one should try to find x_i such that $p(x_i|\mathbf{e}) > V_0/V$. Again we see a belief threshold V_0/V.

To summarize, when making a decision under uncertainty, one often has a utility dependent lower bound p_0 such that one needs to find the composite hypothsis x_i which satisfies $p(x_i|\mathbf{e}) > p_0$. In the about two extreme cases, p_0 takes the values of p' and V_0/V respectively. We call such x_i *Lower Bounded Composite Hypothesis* (LBCH).

LBCHs may not be unique if $p_0 \leq 0.5$.

Definition 2 *A composite hypothesis x is* **dominant** *if and only if $p(x|\mathbf{e}) > 0.5$.*

Proposition 1 *A LBCH is a MLCH and is unique if it is dominant.*

Since dominant LBCHs are profitable, unique and most likely, they are more useful than non-dominant LBCHs.

Given a state of knowledge, there may not exist any LBCH relative to a particular p_0. One can, however, strive for LBCHs by collecting more evidence to increase the amplitude of $p(x_i|\mathbf{e})$.[1]

3 Computing LBCHs with Belief Updating

This section derives the condition under which belief updating can be used to compute LBCHs.

Consider a set $X = A_1 \ldots A_n$ of binary (this restriction will be eliminated latter) variables and an instantiation x of X. The following lemma provides a sufficient condition for computing LBCHs.

Lemma 1 *Let $x = a_1 \ldots a_n$ be an instantiation of a set X of n binary variables. x is a LBCH relative to p_0 if $p(a_i|\mathbf{e}) > (n - 1 + p_0)/n$ $(i = 1 \ldots n)$.*

Proof:

To simplify the notation, write $p(a_1 \ldots a_n|\mathbf{e})$ as $p_{1\ldots n}$, and $p(\overline{a_1}a_2 \ldots a_n|\mathbf{e})$ as $p_{\overline{1}\ldots n}$ where the bar means 'not'. Let $T = (n - 1 + p_0)/n$.

$$
\begin{aligned}
\sum_{i=1}^{n} p_i \;=\; & (p_{1\ldots n} + \underbrace{p_{1\overline{2}\ldots n} + \ldots + p_{12\ldots\overline{n}}}_{C_{n-1}^1} + \\
& \underbrace{p_{1\overline{23}\ldots n} + \ldots + p_{1\ldots\overline{n-1}\,n}}_{C_{n-1}^2} + \ldots + \underbrace{p_{12\ldots\overline{n}}}_{C_{n-1}^{n-1}}) \\
& + \ldots + \\
& (p_{1\ldots n} + \underbrace{p_{\overline{1}\ldots n} + \ldots + p_{1\ldots\overline{n-1}\,n}}_{C_{n-1}^1} + \\
& \ldots + \underbrace{p_{\overline{2}\ldots\overline{n-1}n}}_{C_{n-1}^{n-1}}) \\
= \; & n(p_{1\ldots n}) + \\
& (n-1)(\underbrace{p_{\overline{1}\ldots n} + \ldots + p_{1\ldots\overline{n}}}_{C_n^1}) + \\
& (n-2)(\underbrace{p_{\overline{12}\ldots n} + \ldots}_{C_n^2}) + \\
& \ldots + \\
& 1(\underbrace{p_{\overline{1}\ldots\overline{n-1}n} + \ldots}_{C_n^{n-1}}) \\
> \; & nT
\end{aligned}
$$

Since the terms in the parentheses in the second equation are all different, the above implies

$$p_{1\ldots n} > nT - (n-1) = p_0$$

□

A variable A of multiple values can be transformed into a binary variable A', with the outcome a of A in which one is interested as one outcome of A', and with other outcomes of A congregated as the other outcome of A'. Since such transformation is always possible, Lemma 1 can be generalized to variables other than binary ones.

Lemma 1 establishes a threshold for each $p(a_i|\mathbf{e})$. Once each variable in X has one outcome passing the threshold, the conjunction of this set of outcomes forms a LBCH on X. It can be seen from the above proof that the threshold for each outcome is not necessary. Replacing such threshold by a threshold for the sum of $p(a_i|\mathbf{e})$, one has the slightly stronger result ((2) in the following theorem).

Theorem 1 *Let $x = a_1 \ldots a_n$ be an instantiation of a set X of n variables. x is a LBCH relative to p_0 if*

1. *(individual threshold) $p(a_i|\mathbf{e}) > (n - 1 + p_0)/n$ $(i = 1 \ldots n)$ or*
2. *(sum threshold) $\sum_{i=1}^{n} p(a_i|\mathbf{e}) > n - 1 + p_0$.*

Table 1 shows the threshold for (1) in theorem 1 in relation with n when $p_0 = 0.5$.

n	2	3	5	10	20
(n-0.5)/n	0.75	0.83	0.90	0.95	0.975

Table 1: Threshold for marginal probabilities

4 How Restrictive is the Threshold Method?

Table 1 shows that the computation of LBCH (with $p_0 = 0.5$) of more than about 20 variables by belief updating requires above 0.975 posterior probability for each outcome involved. This seems to be too tight a restriction on the size of the application domain. Fortunately, although the number of variables of an application domain can easily exceed 20, computation of LBCHs over less than about 20 variables can often be justified and can even be desirable. Since dominant LBCHs are also MLCHs (Proposition 1) and are more useful than non-dominant LBCHs, the following discussion refers majorly to MLCHs to match the concept used in the referenced literature. We specify the conclusion applicable to LBCHs whenever possible.

[1] For simplicity of presentation, we assume the cost of collecting evidence is much smaller than $V + W$.

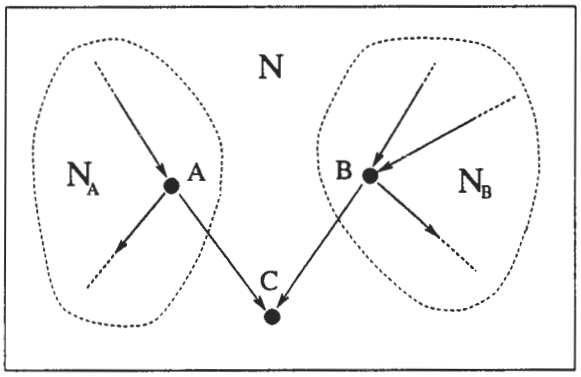


Figure 1: Illustration of problems with belief revision

4.1 Belief revision over all variables is not always desirable

The difficulties associated with belief revision over all variables have been identified in [Pearl 88, Poole and Provan 90]. First of all, there is the computational problem. If the MLCH is to cover the entire domain, every piece of evidence must be propagated to the entire network. Unobserved consequence variables no longer block the propagation as is the case in belief updating. For example (Figure 1), if variable C is unconfirmed, the subnetwork N_A is independent of N_B if PMDs are to be computed. However, in the case of belief revision over all variables, when evidence is available in N_A, it has to be propagated to N_B in order to find the MLCH. This would impose heavy computational overhead in larger applications.

Secondly, there is the conceptual problem. In belief revision, changing the space where MLCH is computed will change the outcome for each variable. For example (Figure 1), assuming all evidence is contained in N_A, and A, C are uncertain, computing the MLCH in N_A or in N may end up with different assignments for variable A. When a diagnosis problem is coupled with a holiday planning, the imagination of the uncertainty of trips can render a highly likely normal diagnosis to become positive (with the disease) as pointed out by Pearl [1988].

It should be emphasized that this conceptual problem is not unique to belief revision in Bayesian networks. Belief revision can be viewed as an optimazation task. Optimization is, in general, sensitive to the space over which some target function is optimized. Belief updating does not suffer from the above conceptual problem because it is not a optimazation task.

The above problems suggest that computation of the MLCH over all variables of the domain is not always desirable.

4.2 Variables which should be covered by a MLCH

In order to overcome the problems associated with belief revision over all variables, Pearl [1988] proposed to circumscribe a domain into a set of variables called *explanation corpus*, and to compute the MLCH only over the corpus. According to Pearl, the corpus includes evidence and ancestors of evidence variables which have a significant impact on pending decisions. Restricting computation of the MLCH on the corpus will eliminate a substantial number of variables in the domain. In this subsection, we take a closer look at the variables included in the corpus. The purpose is to examine the possibility of focusing the computation to an even smaller set of variables.

Typically a known evidence variable takes one of its outcomes. Therefore, within the corpus, one can further eliminate the evidence variables from computation of the MLCH. The remaining variables are the ancestors of the evidence variables. These variables can be organized in different levels of abstractions. For example, in medical applications, a Bayesian net can have symptom variables as leaves. The successive next higher levels can be biological state level, disease or syndrome level, and disease groups level (representing diseases with some common characteristics). Not all of them are of a physician's current interest. Usually, only variables at certain level of abstraction is of physician's utmost interest. The number of such variables will be much smaller than the number of variables within the explanation corpus.

When the number of variables in the abstraction level of current attention is large, an important strategy can be applied as suggested in [Patil, Szolovits and Schwartz 82]. One can move up an abstraction level where the important differences are represented by a smaller set of higher level hypotheses. After gathering further evidence such that the differentiation at the higher level is resolved, one can move down to the lower level to differentiate in greater detail. Focusing one's attention in this fashion, one needs only to compute the MLCH on a small number of variables.

The existence of a LBCH over a set of variables is relative to an adequate amount of evidence. Thus it is often useful to know there is no LBCH over a set of variables given the current state of knowledge. The following theorem gives one condition. Its proof is trivial.

Theorem 2 *Let X be a set of n variables. Let $A \in X$ have m possible outcomes $\{a_1, \ldots, a_m\}$. If $p(a_i|\mathbf{e}) < p_0$ $(i = 1 \ldots m)$, then none of the instantiations of X is a LBCH relative to p_0.*

Theorem 2 asserts that there is no LBCH if A is involved in the composite hypothesis. That is, given the available evidence, one can not make any strong commitment on any composite hypothesis which involves A. Perhaps, one could choose to collect more evidence in order to sharpen the distribution and to make a better informed decision or to make a decision with A ignored. Both can be seen in medical diagnosis. When a doctor does not have adequate evidence to give a positive diagnosis on a disease, he/she may either perform more tests on the patient or give no treatment until the disease further manifests.

5 Computation of LBCHs

Computationally, we specify the abstraction level of current interest in the explanation corpus. To compute

dominant LBCHs, we uses Theorem 2 to exclude the variables at the current level which do not lead to a LBCH. Suppose a set X of M variables remains after the exclusion. Using the threshold in theorem 1, we can find all dominant LBCHs covering a subset of 2 variables in X, 3 variables, ..., up to $n \leq M$ variables, where n increases as more evidence is gathered. The following Algorithm formally presents this method. It can easily be extended to cover the non-dominant LBCH case.

Algorithm 1 *Let X be the set of variables in the explanation corpus and $X' \subseteq X$. X' corresponds to the restricted corpus after the application of Theorem 2. Let $p(A_i|\mathbf{e})$ be the PMD for variable $A_i \in X$. Let p_0 be the lower bound of the posterior probability on dominant LBCHs. Let x^j be a composite hypothesis over a subset of j variables of X'. Let Y be the set of dominant LBCHs (over different subsets of X').*

begin

Set $X' = X$.
For each i, remove A_i from X' if, for every outcome a_{ik} of A_i, $p(a_{ik}|\mathbf{e}) < p_0$.
Set index $j = 1$. Set FLAG on. Set $Y = \{\}$.
While FLAG is on, do the following:
Set FLAG off.
Increase j by 1.
For each x^j, insert x^j to Y and set FLAG on if the threshold in Theorem 1 is satisfied.

end.

6 Computation of Dominant LBCHs in Secondary Structures

Inference in Bayesian nets can be performed in original networks [Pearl 86, Shachter 88, Henrion 88, Pearl 88, Henrion 91] or be performed in clique tree based secondary structures [Lauritzen and Spiegelhalter 88, Jensen, Lauritzen and Olesen 90a, Xiang, Poole and Beddoes 91]. The above threshold method for computing LBCHs can be used in conjunction with algorithms of belief updating in both structures. Evidential reasoning in secondary structures have advantages when the domain knowledge is acquired once and is used for multiple cases (e.g. expert systems); and when the net topology is multiply connected. To the author's knowledge, however, there has been no general algorithm which computes MLCHs in clique tree based secondary structures. All known algorithms for secondary structures perform belief updating and compute PMDs. The threshold method of Theorem 1 allows one to obtain the LBCHs in secondary structures using existing belief updating algorithms.

An alternative for computing MLCHs (or dominant LBCHs) using belief updating is to introduce auxiliary nodes [Pearl 88]. For a set X of variables, an auxiliary node A with all the possible joint assignments xs as its outcomes is added to the original net. The method compute both the MLCH and the corresponding probability value. However, it is difficult to integrate this method with clique tree based secondary structures. For each different X and the corresponding auxiliary node, a recompilation of the original net into a secondary structure is required. Since an auxiliary node introduces possibly several additional loops, the addition can also complicate the secondary structure and increase the computational complexity.

Compared with the auxiliary node method, the threshold method computes only LBCHs and not their probability values. But the method does not require any recompilation of the secondary structure and supports dynamic change of the set of variables on which LBCHs are to be computed. The extra computation beyond that is required by belief updating is trivial.

7 An Example

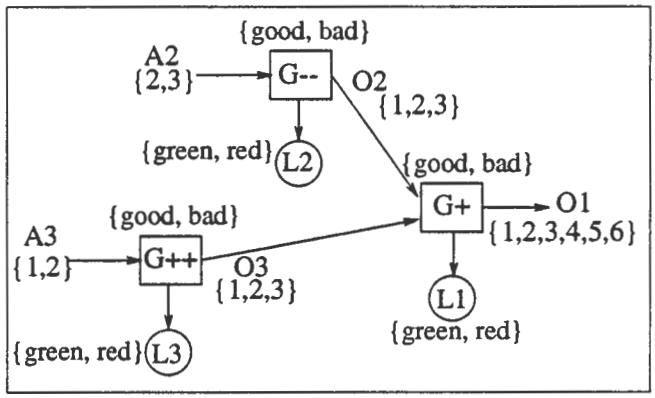


Figure 2: three gates circuit example

This section provides a simple circuit diagnosis example to illustrate the threshold method.

The circuit consists of three gates: G−−, G++, and G+ (Figure 2). G−− decreases its input A_2 by one. G++ increases its input A_3 by one. G+ adds the output O_2, O_3 of G−− and G++ to produce O_1. The domains of input and output of each gate is illustrated in Figure 2. Each gate has a prior probability of being faulty (0.03, 0.02 and 0.01 for G+, G−−, and G++, respectively). When G−− and G++ are faulty, their outputs are clamped to their inputs with 98% of chance, and can be any other values in their output domain with 1% of chance for each value. When G+ is faulty, its output is clamped, with 95% of chance, to one of the two inputs which has a greater value, and can be any other values in its output domain with 1% chance for each value. Suppose one has no access to O_2, O_3, but can observe an indicator light for each gate (L_1 for G+, L_2 for G−−, and L_3 for G++). Each light monitors the status of the corresponding gate with 'red' indicating a faulty state, and 'green' a normal state. The lights are not totally reliable though. When the gate is faulty, only 70% of time, the light turns red. When the gate is normal, there is still 10% of chance the light gives false alarm (red).

The circuit is represented by a Bayesian net in Figure 3. The probabilistic assumptions are summarized below. Histograms show the PMDs with the bottom level being 0 and the top level being 1. Figure 3 to 6 are produced using the *WEBWEAVR* expert system shell [Xiang et al. 91].

$$
\begin{aligned}
p(G+ = bad) &= 0.03 \\
p(G-- = bad) &= 0.02 \\
p(G++ = bad) &= 0.01 \\
p(A2 = 2) &= 0.5 \\
p(A3 = 1) &= 0.5 \\
p(O2 = A2 - 1 | A2, G-- = good) &= 1 \\
p(O2 = A2 | A2, G-- = bad) &= 0.98 \\
p(O2 \neq A2 | A2, G-- = bad) &= 0.01 \\
p(O3 = A3 + 1 | A3, G++ = good) &= 1 \\
p(O3 = A3 | A3, G++ = bad) &= 0.98 \\
p(O3 \neq A3 | A3, G++ = bad) &= 0.01 \\
p(O1 = A2 + A3 | A2, A3, G+ = good) &= 1 \\
p(O1 = max(A2, A3) | A2, A3, G+ = bad) &= 0.95 \\
p(O2 \neq max(A2, A3) | A2, A3, G+ = bad) &= 0.01 \\
p(L = red | G = good) &= 0.1 \\
p(L = red | G = bad) &= 0.7
\end{aligned}
$$

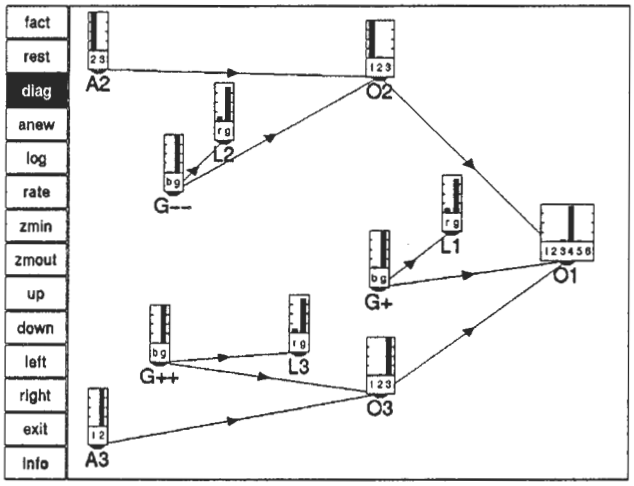


Figure 3: Expected behavior of three gates circuit after inputs A_2, A_3 are specified.

Suppose one knows that the inputs are $A_2 = A_3 = 2$. Figure 3 shows sharp distributions for all gates, which represents one's belief that, without any evidence, everything is expected to be normal. The expected output of the circuit is 4.

Unfortunately, the output is measured as 6. Entering this evidence into the network, PMDs of both G-- and G+ show the possibilities of fault (the outcome 'bad' (b) in both gates has a probability value greater than 0.4 in Figure 4). With the understanding of the circuit, one can tell that only two cases are quite likely. One is that G-- is faulty and G+ and G++ are normal. In this case G-- produces output 3 which explains the error. The other is that G+ is faulty and G-- and G++ are normal. In this case G-- produces the expected value 1 and G+ generates the error. Other combinations are quite unlikely. However, without assuming this global

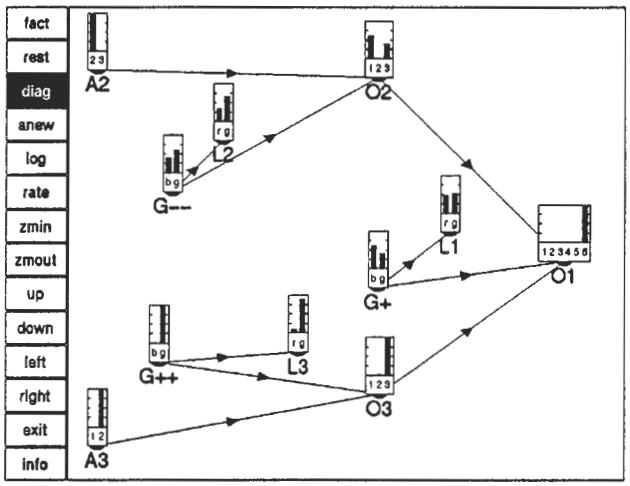


Figure 4: Output $O_1 = 6$ rather than expected 4 is observed.

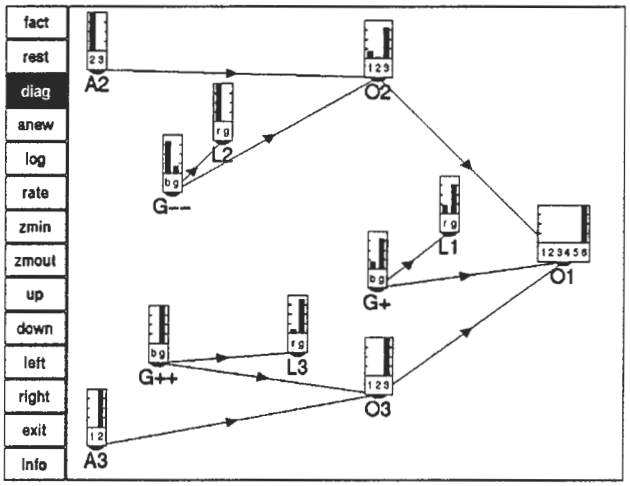


Figure 5: The indicator light L_2 is found to be red (signaling faulty).

knowledge, it is not obvious from the PMDs whether both gates are faulty, or only one of them. Let the lower bound $p_0 = 0.5$. Since none of the outcomes of either gate is greater than 0.75, the threshold by theorem 1, one can not determine the LBCH for this pair of gates. Additional evidence has to be gathered.

After checking the indicator light L_2 and entering the value 'red' (signaling faulty), Figure 5 shows $p(G-- = bad|\mathrm{e}) = 0.82$, $p(G+ = good|\mathrm{e}) = 0.80$, $p(G++ = good|\mathrm{e}) = 0.99$. To determine the LBCH over two variables G-- and G+, this result is sufficient according to the individual threshold of theorem 1. The result is also sufficient for the LBCH over three gate variables according to the sum threshold of theorem 1 (the threshold is 2.5). But it is not sufficient for the LBCH over three gate variables if the individual threshold of theorem 1 is adopted (0.83). Suppose this threshold is used and one gathers another piece of evidence $L_1 = green$

(Figure 6). One finally has $p(G--=bad|\mathbf{e}) = \mathbf{0.93}$, $p(G+=good|\mathbf{e}) = \mathbf{0.92}$, and $p(G++=good|\mathbf{e}) = \mathbf{0.99}$, which give the LBCH over three gate variables by any of the two thresholds. Using auxiliary node method for three gates, the LBCH has the probability value 0.92.

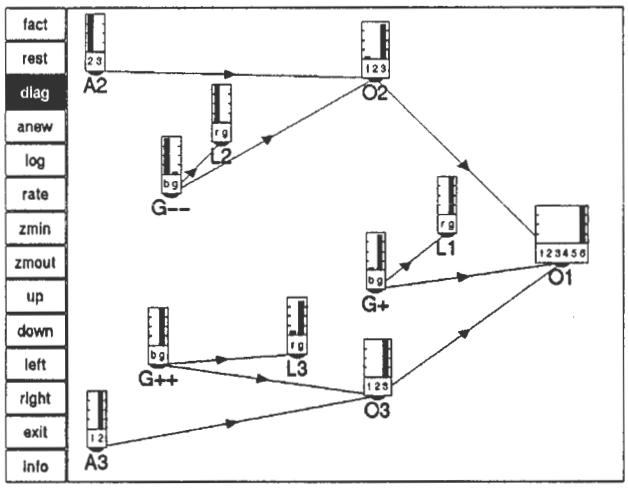


Figure 6: The indicator light L_1 is found to be green (signaling normal).

8 Conclusion

This paper presents a straightforward threshold method of computing the LBCH via belief updating. The concept of LBCH is introduced in a decision theoretic context where the decision maker imposes a threshold on the expected utility. Dominant LBCHs are also MLCHs. The concept can equally well be introduced in the context where utility functions are non-negative. The selected context is felt to be closer to many practical situations.

Although the threshold method is practical only if the number of variables over which the LBCH is computed is small, it is argued that computing the LBCH over small number of variables can often be justified and is often desirable.

The method can be used in conjunction with any belief updating algorithm either based on original networks or based on secondary structures. There has been no known algorithm performing belief revision in clique tree based secondary structures. Such secondary structures have advantages when domain knowledge is acquired once and is used for multiple cases (e.g. expert systems), and when the net topology is multiply connected. Therefore, the threshold method provides secondary structure based algorithms with additional capability of obtaining LBCHs. The extra computation required is trivial.

Some features of the method are worth pointing out. First, the method does not provide probability values of LBCHs. In most cases, the actual probabilities of LBCHs are much larger than lower bounds such that high expected utility are associated with the corresponding decisions. Second, before the adequate evidence is gathered such that a LBCH can be warranted, no LBCH can be given. When users have the option to collect more evidence, this feature prompts users not to stop with premature judgments. However when users are bounded with limited resource (time or information sources), it is possible that no LBCH is provided even if the actual MLCH has a probability greater than the lower bound. This is because the method is based on a sufficient condition of LBCHs (Theorem 1) which is not a necessary condition. It seems this is not a major problem since users can always rely on PMDs for rough guidance.

Acknowledgements

This research is supported by Operating Grant OGP0090307 from NSERC, and CRD3474 from the Centre for Systems Science at Simon Fraser University. The author would like to thank David Poole, Nevin Zhange, Bill Havens and Stefan Joseph for insightful comments to the early manuscript.

References

[Andreassen et al. 89] S. Andreassen, F.V. Jensen, S.K. Andersen, B. Falck, U. Kjerulff, M. Woldbye, A.R. Sorensen, A. Rosenfalck and F. Jensen, "MUNIN - an expert EMG assistant", J.E. Desmedt, (Edt), *Computer-Aided Electromyography and Expert Systems*, Elsevier, 255-277, 1989.

[Heckerman 90] D. Heckerman, A tractable inference algorithm for diagnosing multiple diseases, M. Henrion, R.D. Shachter, L.N. Kanal and J.F. Lemmer. (Eds.), *Uncertainty in Artificial Intelligence 5*, Elsevier Science Publishers, 163-171, 1990.

[Henrion 88] M. Henrion, Propagating uncertainty in Bayesian networks by probabilistic logic sampling, J. F. Lemmer and L. N. Kanal (Eds), *Uncertainty in Artificial Intelligence 2*, Elsevier Science Publishers, 149-163, 1988.

[Henrion 91] M. Henrion, Search-based methods to bound diagnostic probabilities in very large belief nets, B.D. D'Ambrosio, P. Smets and P.P. Bonissone (Eds.), *Proc. 7th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, 1991.

[Huchlenbroich 91] P. Huchlenbroich, Methodological principles in medical knowledge programming: part 1, *Artificial Intelligence in Medicine*, Vol.3, No.2, 113-123, 1991.

[Jensen, Lauritzen and Olesen 90a] F.V. Jensen, S.L. Lauritzen and K.G. Olesen, Bayesian updating in causal probabilistic networks by local computations, *Computational Statistics Quarterly*, 4, 269-282, 1990.

[Jensen, Olesen and Andersen 90b] , F.V. Jensen, K.G. Olesen and S.K. Andersen, "An algebra of Bayesian belief universes for knowledge based systems", *Networks*, Vol. 20, 637-659, 1990.

[Lauritzen and Spiegelhalter 88] S.L. Lauritzen and D.J. Spiegelhalter, Local computation with probabilities on graphical structures and their

application to expert systems, *Journal of the Royal Statistical Society*, Series B, 50, 157-244, 1988.

[Patil, Szolovits and Schwartz 82] R.S. Patil, P. Szolovits and W.B. Schwartz, "Modeling knowledge of the patient in acid-base and electrolyte disorders", P. Szolovits (Edt), *Artificial Intelligence in Medicine*, 191-226, 1982.

[Pearl 86] J. Pearl, "Fusion, propagation, and structuring in belief networks", *Artificial Intelligence*, 29: 241-288, 1986.

[Pearl 88] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988.

[Poole and Provan 90] D. Poole and G.M. Provan, What is an optimal diagnosis?, *Proc. Sixth Conference on Uncertainty in Artificial Intelligence*, Cambridge, Mass., 46-53, 1990.

[Shachter 86] R.D. Shachter, Evaluating influence diagrams, *Operations Research*, Vol.34, 6, 871-882, 1986.

[Shachter 88] R.D. Shachter, Probabilistic inference and influence diagrams, *Operations Research*, Vol.36, 4, 589-604, 1988.

[von Neumann and Morgenstern 47] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton University press, Princeton, NJ, 1947.

[Xiang et al. 91] Y. Xiang, B. Pant, A. Eisen, M.P. Beddoes, and D. Poole, PAINULIM: a neuromuscular diagnostic aid using multiply sectioned Bayesian belief networks, *Proc. ISMM International Conference on Mini and Microcomputers in Medicine and Healthcare*, Long Beach, CA, 64-69, 1991.

[Xiang, Poole and Beddoes 91] Y. Xiang, D. Poole, and M. P. Beddoes, Multiply sectioned Bayesian networks and junction forests for large knowledge based systems, submitted to *Computational Intelligence*, 1991.

A Rational Agent Can Be Surprised No Matter What

Yen-Teh Hsia*
IRIDIA, Université Libre de Bruxelles
50 av. F. Roosevelt, CP 194/6
1050, Brussels, Belgium

Abstract

We identify an abstract situation in which one can strongly argue for the rationality of being surprised no matter whether some particular proposition is true, and we show that we cannot use probability theory to capture the kind of intuitions that we want to formalize. A belief-function solution is described.

1 Introduction

Let us assume that we, as humans, are capable of being surprised, and that our intuitive degrees of surprise can also be measured in some way.[1] Having made these two assumptions, we are interested in the following questions.

1. Are there situations in which a rational agent would be surprised no matter what?
2. If there are such situations, how can we formalize it?

Here, by "a rational agent," we mean an agent who tries to be as objective as it/he/she can in making subjective judgments. And by "being surprised no matter what," we mean the situation is such that the agent would be surprised upon realizing that some proposition A is true, and the agent would also be surprised upon realizing that A is false.

The first question is actually the subject of a long time debate. G. L. S. Shackle, for example, wrote [Shackle 61, p. 75]:

> To assign greater than zero degrees of potential surprise to both the hypothesis and its contradictory would ... betray an unresolved mental confusion.

Glenn Shafer [76, pp. 225-6] disputes Shackle's contention, arguing that "the occurrence of outright conflict in our evidence" may nevertheless result in mental states in which we assign greater than zero degrees of potential surprise to both a hypothesis and its contradictory. Isaac Levi [84; pp. 223-7] disagrees with Shafer, and wrote [p. 226]:

> My contention is that ... it is not to be found in recommending that rational men believe both hypotheses to a positive degree.

Our own perspective is this. Though we perfectly agree that, in general, we are not (and possibly should not be) surprised no matter what, there may nevertheless be circumstances in which we would be surprised no matter whether some particular proposition is true. The purpose of this paper, then, is to identify one such circumstance along with its associated rationality. Once we have done so, it is then desirable that we answer the second question as well. In fact, this second question is interesting in its own right, because the prevailing view seems to be that we SHOULD BE ABLE TO use probability theory to capture the kind of intuitions that we want to formalize. As it turns out, probability theory will not work. Nor does possibility theory in the sense of Shackle [49, 61] or Zadeh [78].

This paper is organized as follows. First we identify one circumstance under which a rational agent would be surprised no matter what (Section 2). We then show that probability theory cannot be used to formalize the kind of intuitions that we want to capture, and we show how we can use belief functions to do it (Section 3). Section 4 concludes.

2 Permissibility of being surprised no matter what

We consider the following framework. Let $\mathfrak{P}$ be a finite, non-empty set of propositional primitives, Θ be the set of all (total) valuations of $\mathfrak{P}$, and $\mathfrak{L}_\mathfrak{P}$ be the least set of formulas containing $\mathfrak{P}$, closed under $\neg$, $\wedge$. Call any subset of Θ a *proposition*, and for every formula β in $\mathfrak{L}_\mathfrak{P}$, we let $[\beta]$ be the set of models of β (valuations under which β is true). We assume that *the actual situation* or *the actual state of affairs* is uniquely represented by an element of Θ. That is to say, if we are able to find out about everything that we need to know about the actual situation, then we can

* This work was supported in part by the DRUMS project funded by the Commission of the European Communities under the ESPRIT II-Program, Basic Research Project 3085.

[1] The author has proposed a standard for measuring our degrees of surprise in [Hsia 91a]. The proposed standard (called a canonical measurement device) is not as easy to use as we would like it to be, but it seems to be acceptable in theory. This measurement is now described in the appendix.

ALWAYS identify exactly one element of Θ as THE representation of the actual situation. And thus, by saying that we would be surprised that a proposition A is true (or a formula β is true, or the proposition $[\beta]$ is true), it just means that we EXPECT to find the representation of the actual situation in $\Theta \backslash A$ (or $[\neg\beta]$). As a convention, we use $S(A)$, where $0 \leq S(A) \leq 1$, to denote our degree of potential surprise that A is true; $S(A) = 0$ means "I would not be surprised upon realizing that A is true," and $S(A) = 1$ means "I would be totally surprised upon realizing that A is true." We also accept a notion of conditional potential surprise here, and we use $S(. \mid B)$ to denote our degree of potential surprise GIVEN that B is true. This makes $S(A)$ an abbreviation of $S(A \mid \Theta)$.

The question we are asking, then, is this. Can we identify a certain situation such that, when formulated in the above framework, it is rational to have $0 < S([\beta]), S([\neg\beta]) < 1$ for some formula β? To give a positive answer to this question, we need to describe what this situation is, what β is, and even more importantly, why it is rational to have $0 < S([\beta]), S([\neg\beta]) < 1$. This is what we do in this section. Below, we first describe what we consider to be a basic rationality of surprise. Then, we motivate the thought that we can be rational while being surprised no matter what. This is followed by an abstract description of a particular situation, along with two real-life examples.

We offer the following rationality of surprise.

We would be surprised that β is true WHENEVER we have a reason for expecting β to be false.

Here, the notion of "reason" (for expecting β to be false) is deliberately left undefined; it is something that is subject to our judgments, and this judgment is based on two things: (1) our general understanding of how things work (i.e., background knowledge), and (2) our information at hand about the actual situation (i.e., the context). To illustrate what we mean by "a reason for expecting something," consider the following examples. Suppose there is a random device that generates event-1 with .9999999 chance and event-2 with .0000001 chance. Then we might say there is a good reason for expecting the random device to generate event-1, since there is almost a sure chance that event-1 will occur.[2] However, if the random device is such that it generates event-1 with .5 chance and event-2 with .5 chance, then (so it seems) there is no reason why we should expect the random device to generate event-1 (or event-2), since the two events are equally likely to occur. Finally, suppose we only know that the device generates either event-1 or event-2 but we have no idea whether the device is random (not to mention about "the" chance set up). Then there does not seem to be any reason why we should expect the device to generate event-1 (or event-2), since we are equally uncertain about both events.

Now that we have established a link between surprise (that something is true) and reason (for expecting the contrary), the question we want to answer just amounts to the following. Are there situations in which we have a legitimate reason for expecting some β to be true and we also have a legitimate reason for expecting β to be false? In the very least, the answer to this question does not seem to be very straightforward. Consider, for example, the now famous Nixon story. We are faced with the following situation. Given (the only information) that Nixon is a republican, we would be surprised that he is a pacifist. And given that Nixon is a quaker, we would be surprised that he is not a pacifist. Now we are given the information that Nixon is both a republican and a quaker, would we be surprised that he is a pacifist (as he is a republican), and would we be surprised that he is a non-pacifist (as he is a quaker)? But even more importantly, SHOULD we consider ourselves irrational IF we would be surprised no matter whether Nixon is a pacifist? This, of course, depends on whether we consider being a republican (alternatively, a quaker) a "legitimate reason" for expecting him to be a non-pacifist (alternatively, a pacifist), despite the fact that he is also a quaker (alternatively, a republican). We do not think there is a clear cut answer here. As a second example, consider cases in which we have two credible witnesses testifying under oath, and they are contradicting each other. We know one of them is not telling the truth. But the question is: should we ALWAYS completely suppress our faith in at least one of the two witnesses? Clearly, if we do not do so, then we would be (and are certainly entitled to be) surprised no matter whether witness-1 is telling the truth.[3] But why should we always completely suppress our confidence in at least one witness?

By now, we hope to have raised some interest in the issue of being surprised no matter what. Next, let us try to identify a situation in which one could strongly argue for the rationality of being surprised no matter what. The basic idea is as follows. Suppose there are two primitives X and Y ($\mathcal{P} = \{X, Y\}$). We have a reason (call it R1 for convenience) for expecting X to be false, i.e., $S([X]) > 0$. We also have a reason R2 for expecting Y to be false, i.e., $S([Y]) > 0$. These two reasons are intuitively independent in the following sense : whatever we learn about X (alternatively Y), it will have no effect on our judgments that R1 (alternatively, R2) constitutes a legitimate reason for expecting Y (alternatively, X) to be false. Thus, for example, X can be "friend-1 has spoken out against you," Y can be "friend-2 has spoken out against you," while R1 is that friend-1 has been a good friend of yours since high school, and R2 is that friend-2 has been a good friend of yours ever since you worked together in the same company. Say that you consider R1 and R2 to be independent in having an impact on your judgment; that is, whether or not friend-1 (alternatively, friend-2) has spoken out against you, it will have no effect on your judgment that R2 (alternatively, R1) constitutes a reason for expecting Y (alternatively, X) to be false. Now suppose you learn that $X \vee Y$ is true. That is, you realize that at least one of friend-1 and friend-2 has spoken out against you. We argue that, at

[2] But note that it is entirely up to one's OWN judgment as to whether something constitutes a reason for expecting some proposition to be true.

[3] If witness-1 is not telling the truth, then it would be somewhat surprising. But if witness-1 is telling the truth, then witness-2 must be telling a lie. This would be somewhat surprising also.

this point, R1 (alternatively, R2) still constitutes a reason for expecting X (alternatively, Y) to be false, even though its impact on our judgments is now somewhat weakened by the revelation that $X \vee Y$ is true. In other words, we are saying that we should still have $S([X] \mid [X \vee Y]) > 0$ (because of R1) and $S([Y] \mid [X \vee Y]) > 0$ (because of R2), though we also think that $S([X] \mid [X \vee Y])$ and $S([Y] \mid [X \vee Y])$ should be strictly less than $S([X])$ and $S([Y])$, respectively. After all, R1 and R2 are STILL there in our background considerations, despite the fact that $X \vee Y$ is true. As we consider them to be independent in having an impact on our judgments, we do not see how we can "disqualify" any one of them upon realizing that $X \vee Y$ is true. Because the information that $X \vee Y$ is true is absolutely NON-INFORMATIVE in this respect; i.e., it does not tell us anything about the "illegitimacy" of R1, nor does it say anything about the illegitimacy of R2. As a result, we have no choice but to enter an epistemic state of confusions.[4]

A more complete example may help to illustrate the point.

Example 1. (Chemical plant) The three-member city council is now meeting behind closed doors to decide about the fate of a proposed plan for building a chemical plant. Council member A comes from District-1 where unemployment rate is extremely high (say, 50 %), and has announced that he/she will vote "yes". Council member B comes from District-2 where unemployment rate is also extremely high, and has also announced that he/she will vote "yes". Council member C, however, comes from District-3 where unemployment rate is low, and has announced that he/she will vote "no" for fear of pollution, despite the fact that there is a fairly reasonable effort in the plan to prevent possible contaminations. YOU are the president of the group called "Citizens for the plant". Members of this group reside mainly in District-1 and District-2, and are all friends or relatives of yours. You have lobbied very hard for the approval of the plan, and have got written promises from both A and B to vote "yes".

Now the waiting is over, and C comes out and happily announces that the vote is two against and one for (surprise!); the plan is now officially dead. But when asked who else voted against the proposal, C just says "you have to ask them yourselves," and leaves. While waiting very eagerly for A and B to come out from behind the door (where they are obviously quarreling; but you cannot hear a word), a being by the name of RATIONALITY approaches you and asks: would you be surprised if it is A who voted "no"? And would you be surprised if it is B who voted "no"?

To analyze this example, we first identify its components as follows.

X : A voted "no"

Y : B voted "no"

Reason for expecting $\neg X$ (i.e., R1) : Unemployment rate in District-1 is extremely high. Besides, A has given his/her promise, in writing, to vote "yes".

Reason for expecting $\neg Y$ (i.e., R2) : Unemployment rate in District-2 is extremely high. Besides, B has given his/her promise, in writing, to vote "yes".

We contend that it is perfectly rational that we be surprised no matter whether X is true (if X is false, then Y must be true), even though we know very well that $X \vee Y$ is true. Because, despite the fact that $X \vee Y$ is true, R1 still constitutes a reason (a very good one, in fact) for expecting X to be false, and R2 still constitutes a reason for expecting Y to be false. We simply do not think we should COMPLETELY IGNORE the impact of these two background reasons in making our judgments.

Below, we give one other such example along with its associated rationalities. As the reasoning with this example is essentially the same, we will not go through it in detail. Instead, we just identify what X, Y, R1, and R2 are in this example.

Example 2. (UFO) You have heard about UFO (Un-identified Flying Object) sightings. However, you simply do not believe that space aliens exist and are visiting earth. "I would be surprised if space aliens do exist and are visiting earth, and I would not be surprised if there are no such things," you said. On the other hand, you also believe that the alleged UFO sightings are actually images of something else, e.g., a balloon or a reflection of the moon. "Nature can play incredible tricks with our eyes. However, I would be surprised if nature is also capable of displaying a realistic, high-tech phenomenon of a UFO," you said. Now you are looking at it - a thing that clearly looks like a flying saucer with three lights on it, and the situation (place, weather, time of the day, etc.) is such that you do not consider your "balloon theory" or "moon image theory" (or whatever other theory you have come up with) applicable in explaining what it is. Now the question is : would you be surprised if UFOs do exist? And would you be surprised if nature is indeed capable of displaying a realistic, high-tech UFO phenomenon?

Analysis of the UFO example.

X : UFOs do exist.

Y : Nature is capable of displaying a realistic, high-tech UFO phenomenon.

Reason for expecting $\neg X$: "There are no such things as space aliens visiting earth!"

Reason for expecting $\neg Y$: Our understanding of how nature works in displaying illusions.

3 Formalizing the intuition of being surprised no matter what

Having identified an abstract situation in which we can strongly argue for the rationality of being surprised no matter what, we now want to see how we might formalize it. In essence, what is required is as follows. We need to find some measure $\mathfrak{M}$ on the space $\Theta_{XY} = \{T_XT_Y, T_XF_Y,$

[4] A related question, then, is how we can be confused while being rational at the same time. We think our subsequent treatment of this X-Y problem could shed some light on this point.

$F_XT_Y, F_XF_Y\}$ such that $\mathfrak{M}$ formalizes the idea that $S([X] \mid [X \vee Y]) = S(\{T_XT_Y, T_XF_Y\} \mid \{T_XT_Y, T_XF_Y, F_XT_Y\}) > 0$, and $\mathfrak{M}$ also formalizes the idea that $S([\neg X] \mid [X \vee Y]) = S(\{F_XT_Y, F_XF_Y\} \mid \{T_XT_Y, T_XF_Y, F_XT_Y\}) = S(\{F_XT_Y\} \mid \{T_XT_Y, T_XF_Y, F_XT_Y\}) > 0$.

We consider three contenders : (Bayesian) probability theory, possibility theory, and belief functions. Clearly, we can use belief functions in the sense of [Hsia 91a] to do it, because Hsia directly interprets $Bel([\neg X] \mid [X \vee Y])$ as $S([X] \mid [X \vee Y])$, and it is possible to have both $Bel([\neg X] \mid [X \vee Y]) > 0$ and $Bel([X] \mid [X \vee Y]) > 0$. Later, we will give a more detailed account of how this is done. We cannot use possibility theory, however, as long as we interpret $N([\neg X] \mid [X \vee Y])$ - the necessity measure - as $S([X] \mid [X \vee Y])$. This is because with possibility theory, we must have $N([\neg X] \mid [X \vee Y]) = 0$ or $N([X] \mid [X \vee Y]) = 0$. But how about probability theory? We now show that THERE IS NO PROBABILITY DISTRIBUTION P ON Θ_{XY} THAT IS CAPABLE OF FORMALIZING THE IDEA THAT $S([X] \mid [X \vee Y]) > 0$ and $S([\neg X] \mid [X \vee Y]) > 0$.

We consider the following set up. Suppose there is a random device (e.g., an urn containing black balls and white balls) that generates event-1 with chance c and event-2 with chance 1-c. Of course we can assess our degree of belief that the device will generate event-1 (alternatively, event-2), and we submit that it is perfectly rational to assess our degrees of belief as the following probability P : P(event-1) = c and P(event-2) = 1-c.[5] We also submit that whatever value c takes, we ought NOT be surprised in both cases. In other words, we claim that, as far as this random device is concerned, we should have S(event-1) = 0 or S(event-2) = 0. The reason is very simple : either $c \geq .5$, in which case we have NO reason for expecting event-2 to occur (and we leave the issue of whether there IS a reason for expecting event-1 to occur to the individual agent), or $c < .5$, in which case we have NO reason for expecting event-1 to occur. In more formal terms, let X be the proposition that "the random device will generate event-1." What we claim, then, is that we SHOULD have $S([X]) = 0$ or $S([\neg X]) = 0$, WHATEVER c is.

Now we are in a very delicate situation. We just made the tacit assumption that our subjective probability CAN BE used as a way of representing our intuitive notion of surprise. But no matter how we use subjective probability to represent surprise (see, for example, the proposals of Bartlett [52], Gärdenfors [88, p. 168], Good [56], Pearl [88, p. 33] and Weaver [48]), we certainly do NOT mean that we should also take the DOMAIN on which we assess our subjective probability and degrees of potential surprise into account. In other words, by saying that probabilities can represent surprise, we are committed to saying that probabilities ALONE will do the job. To put it another way, given ANY probability distribution P on a variable (a propositional primitive) X, we should always be able to compute $S([X])$ and $S([\neg X])$ without having to know WHAT X intuitively means and WHY we assess P. But clearly, X may well be the above proposition that "the random device will generate event-1," and P may well be the same as the underlying chance set up. Thus, we are led to the following observation. IF we admit there is something called a random device (that works according to some objective probability), and IF our subjective belief is the same as the objective probability when the latter exists and is known to us, and IF we consider it to be irrational to be surprised no matter what in the case of a two-event random device with known probability, THEN we have no choice but to compute a probability-induced measure of surprise S', where $S'([X]) = 0$ or $S'([\neg X]) = 0$, whenever we are given a probability P on a binary variable X. In other words, unless we reject any of the three premises we listed above, we cannot say there is a subjective probability $P(. \mid [X \vee Y])$ on Θ_{XY} that is capable of formalizing the idea that $S([X] \mid [X \vee Y]) > 0$ and $S([\neg X] \mid [X \vee Y]) > 0$.

Next, we show how we can use belief functions [Shafer 76; Smets 88] in the sense of [Hsia 91a] to formalize our intuitions that $S([X] \mid [X \vee Y]) > 0$ and $S([\neg X] \mid [X \vee Y]) > 0$.[6] The basic idea here is NOT to interpret Bel(A) - the belief-value of A - as our "degree of belief that A is true", but to interpret it as our "degree of confidence in having the belief/expectation that A is true" [Hsia 91a; Hsia 92]. To see what the difference is, consider the following example. Someone tosses a fair coin, and we are interested in what the outcome is. From the perspective of "degree of belief," it is perfectly rational to give a .5 degree of belief to the proposition that Head is true and a .5 degree of belief to the proposition that Head is false. But from the perspective of "degree of confidence in having a belief/expectation," however, the rational thing to do here is NOT to have the expectation (and thus the belief) that Head is true (or false). After all, both head and tail have the same chance of being true. Therefore there is no reason for expecting Head to be true (or false). For more discussions on the difference between these two views of belief, see [Hsia 92].

Having adopted the above interpretation of Bel(A), we can now view $Bel(\Theta \backslash A)$ as S(A), as long as we accept the postulate that our potential degree of surprise is PROPORTIONAL to our degree of confidence in expecting the contrary [Hsia 91a]. This gives rise to the following formalization of our intuitions. First, we expect X to be false with confidence α, and we also expect Y to be false with confidence β. This amounts to the following two marginal belief functions Bel_X (defined on $\Theta_X = \{T_X, F_X\}$) and Bel_Y (defined on $\Theta_Y = \{T_Y, F_Y\}$), where $m_X(\{F_X\}) = \alpha$, $m_X(\{T_X, F_X\}) = 1 - \alpha$, $m_Y(\{F_Y\}) = \beta$ and $m_Y(\{T_Y, F_Y\}) = 1 - \beta$ ($0 < \alpha < 1$ and $0 < \beta < 1$). And as we consider our two expectations to be independent of each other, we also have the following constraints that must be satisfied by the underlying joint belief function Bel on the Θ_{XY} space : $Bel([\neg X]) = Bel([\neg X] \mid [Y]) = Bel([\neg X] \mid [\neg Y]) = \alpha$; $Bel([X]) = Bel([X] \mid [Y]) = Bel([X] \mid [\neg Y]) = 0$; $Bel([\neg Y]) =$

[5] This is, in effect, an instance of the so-called Hacking principle [Hacking 75]. Essentially, what it says is the following. If there is an underlying objective probability which is known to us and we know of nothing else, then our belief should be the same as the objective probability.

[6] See the appendix for a short introduction of belief functions.

$Bel([\neg Y] \mid [X]) = Bel([\neg Y] \mid [\neg X]) = \beta$; $Bel([Y]) = Bel([Y] \mid [X]) = Bel([Y] \mid [\neg X]) = 0$. By a theorem in [Hsia 91b], Bel must be $Bel_X^{\uparrow XY} \oplus_{XY} Bel_Y^{\uparrow XY}$, where $Bel_X^{\uparrow XY}$ is the cylindrical extension of Bel_X to Θ_{XY} (e.g., $m_X(\{F_X\}) = \alpha$ leads to $m_X^{\uparrow XY}(\{F_X T_Y, F_X F_Y\}) = \alpha$), $Bel_Y^{\uparrow XY}$ is the cylindrical extension of Bel_Y to Θ_{XY}, and $\oplus_{XY}$ is Dempster's combination on the Θ_{XY} space. Having obtained Bel, our joint belief on Θ_{XY}, we just use Dempster's rule of conditioning for making inferences, and this is what we do when we learn that $X \vee Y$ is true. As a result, $Bel([X] \mid [X \vee Y]) = \beta(1-\alpha) / (1 - \alpha\beta)$ and $Bel([\neg X] \mid [X \vee Y]) = \alpha(1-\beta) / (1 - \alpha\beta)$; that is, we would be surprised no matter whether X is true.[7]

4 Conclusion

One aspect of the existing notion of potential surprise, as was introduced by Shackle [49, 61] and reinforced by Levi [84], is that a rational agent should NOT be surprised no matter whether a proposition is true. In general, this does seem to conform to our intuitions. However, the purpose of this paper is to make the following point. It is POSSIBLE that we find ourselves in a situation in which we would be surprised no matter whether some particular proposition is true; moreover, there can be a clear-cut rationality in that.

We also showed that Bayesian probability theory cannot be used to formalize the idea of being surprised no matter what. This result has an interesting consequence. That is, if our argument is acceptable, then it means that a Bayesian MUST never contemplate on the idea that a rational agent can be surprised no matter what.

Finally, by showing that belief functions may be used to capture the idea of being surprised no matter whether some particular proposition is true. We hope to have demonstrated an important difference between belief functions and Bayesian probability theory.

Acknowledgements

The author is indebted to Philippe Smets and Paul Snow for invaluable comments, and has benefited from discussions with Bob Kennes, Vic Poznanski, Alex Saffiotti, and Michael Wong. Paul, in particular, motivated the writing of this paper. The author also thanks two anonymous referees for pointing out the need to talk about Bayesian probability theory.

Appendix A - A canonical way of measuring our degrees of surprise

The basic idea is that we first provide the user with a simple statistic of how a lottery machine containing an unspecified number of black/white balls "behaves" (e.g., for 10426 trial runs, the outcome is a black one, and for 10576 trial runs, a white one), and then we ask the user to assess his/her intuitive degree of surprise upon realizing that the actual ratio between the number of black balls and the number of white balls is M versus N, where M > N and there is no common deviser. We also ask that the user specify his/her intuitive degree of surprise as a number between 0 and 1, where 0 is defined as the user's intuitive degree of surprise upon realizing that the actual ratio is 1 versus 1 (i.e., the lottery machine contains N black ones and N white ones), and 1 is defined as the user's intuitive degree of surprise upon realizing that the actual ratio is one billion versus 1.

This, theoretically speaking, would allow us to "calibrate" anyone's intuitive degrees of surprise. Once we have made this calibration, we can then use it to measure this very same person's degrees of surprise in any domain. For example, given the only information that the entity we are interested in is a bird, the extent to which we will be surprised by the new information that the entity does not fly may be (judged by us to be) the same as the extent to which we are surprised by the "canonical answer" that the actual ratio is "51 versus 43." If, in calibrating our intuitions, we recorded the extent of our surprise associated with "51 versus 43" as .4, then the extent to which we will be surprised by the new information that the entity does not fly (given the only information that it is a bird) is measured .4. This measurement scheme is clearly subjective, as the extent of surprise associated with "x versus y" (or "not-fly given is-bird") is, in general, different for different people.

Appendix B - A gentle introduction to belief functions

To formalize the notion of having a belief with a certain degree of confidence, we propose the use of the following definition.

Definition. (Belief function) Given a finite non-empty set $\mathcal{P}$ of propositional primitives. Θ, the frame of discernment, is the set of all (total) valuations of $\mathcal{P}$. A *belief function* on Θ is a function Bel: $2^\Theta \rightarrow [0, 1]$ which is characterized by an *m-value function* m_{Bel} (written as "m" whenever confusions can be avoided; m is also called "the m-values of Bel"), where m: $2^\Theta \rightarrow [0, 1]$ satisfies two conditions

(1) $m(\varnothing) = 0$, and

(2) $\sum_{A:\, A \subseteq \Theta} m(A) = 1$;

and for every subset B of Θ, Bel(B) is defined as $\sum_{A:\, A \subseteq B} m(A)$.[8] A subset S of Θ is called a *focal element* of Bel if $m(S) > 0$. When Bel is such that $m(\Theta) = 1$, Bel is called the *vacuous belief function*.

[7] Note that $S([X] \mid [X \vee Y]) = \alpha(1-\beta) / (1 - \alpha\beta) < \alpha = S([X])$, and $S([Y] \mid [X \vee Y]) = \beta(1-\alpha) / (1 - \alpha\beta) < \beta = S([Y])$. That is, our confidence in expecting $\neg X$ or $\neg Y$ has been reduced, as it ought to be the case.

[8] This definition is consistent with [Shafer 76]. Smets [88] has a slightly more general definition called an "open world" definition. In this definition, $m(\varnothing)$ may or may not be 0, as Θ is not assumed to be exhaustive, and Bel(A) is defined as the sum of the m-values of those *non-empty* subsets of A.

The idea is this. Let us say that the reason that we are surprised to various extents in various circumstances (and that sometimes we are not surprised at all) is because we intuitively *decompose* our overall confidence in some way and *commit* the various portions of confidence to various propositions. Given this idea of decomposing and committing confidence, the definition above just pushes it to the extreme, admitting *any* possible decomposition of our overall confidence. For historical reasons, we call this possible decomposition "the m-values."

Next, we need a concept of conditioning.

Definition. (Dempster's rule of conditioning) Let Bel be a belief function on Θ and m be its associated m-values. Let B be a non-empty subset of Θ such that $Bel(\Theta\backslash B) \neq 1$. m(. | B), the m-values of Bel(. | B) (read as *Bel conditioned by B*), is defined as follows :

$\forall C \subseteq \Theta$, **if** $C \subseteq B$

then $m(C \mid B) = \sum_{D:\, D \subseteq \Theta\backslash B} m(C \cup D) / K$

else $m(C \mid B) = 0$,

where $K = 1 - Bel(\Theta\backslash B)$ is the normalization constant.

(Note that for every subset S of Θ, $Bel(S \cap B \mid B) = Bel(S \mid B)$, whereas in general, $m(S \cap B \mid B) \neq m(S \mid B)$.)

The intuition behind this rule is as follows. Let m (i.e., an m-value function) be how we intuitively decompose our overall confidence. Suppose we now receive some information about the actual situation, saying that the truth is in the set B ($B \subseteq \Theta$). Given that the truth is in B, rationality requires that we expect the truth to be in B with total confidence, and that we do not expect the truth to be outside of B. This means that we must cancel any portion of confidence that was originally committed to a subset of $\Theta\backslash B$, and that we redistribute these portions ($Bel(\Theta\backslash B)$, in fact) in some way. What Dempster's rule does then is to redistribute $Bel(\Theta\backslash B)$ in a way that is similar to how Bayes' rule does it - by proportions. But what about the portion of confidence that was originally committed to a set A that has a non-empty intersection with B? As it just *happens* that zero or more elements of A are ruled out in this particular instance, we let the ones that are *not* ruled out "inherit" the portion of confidence that was originally committed to A.

Next, Dempster's rule of combination ($\oplus_h$) - a purely syntactical operation in our framework.

Let h be a non-empty subset of $\mathcal{P}$. Let Bel_1 and Bel_2 be two belief functions on Θ_h (and let m_1 and m_2 be their respective m-values). $Bel_1 \oplus_h Bel_2$ is defined to be the following belief function Bel on Θ_h:

$\forall S \subseteq \Theta_h$, $m_{Bel}(S) =$

$$\sum_{A,B:\, A \cap B = S} m_1(A) m_2(B) / K,$$

where $K = \sum_{A,B:\, A \cap B \neq \emptyset} m_1(A) m_2(B)$.

Finally, we need the notion of belief projection and extension. Let h and g be two subsets of $\mathcal{P}$ such that $h \subseteq g$ (g is $\mathcal{P}$ by default).

The *projection* of an element x of Θ_g (e.g., $g = \{X_1, X_2, X_3, X_5, X_7\}$ and $x = \langle a1, a2, a3, a5, a7 \rangle$) to Θ_h, denoted as $x^{\downarrow h}$, is simply this element with the extra coordinates dropped (e.g., $h = \{X_1, X_3, X_5\}$ and $x^{\downarrow h} = \langle a1, a3, a5 \rangle$).

The *projection* of a subset A of Θ_g to Θ_h, denoted as $A^{\downarrow h}$, is $\{x^{\downarrow h} : x \in A\}$.

Let Bel be a belief function on Θ_g (and let m be its associated m-values), *the projection (or marginalization) of m to h*, denoted as $m^{\downarrow h}$, is defined as

$$m^{\downarrow h}(A) = \sum_{B:\, B^{\downarrow h} = A} m(B).$$

$Bel^{\downarrow h}$ (*the projection of Bel to h*) is, as usual, characterized by $m^{\downarrow h}$.

The *extension* of S ($S \subseteq \Theta_h$) to Θ_g, denoted as $S^{\uparrow g}$, is the set $\{x: x \in \Theta_g \text{ and } x^{\downarrow h} \in S\}$.

Let Bel be a belief function on Θ_h (and let m be its associated m-values), *the extension of m to g*, denoted as $m^{\uparrow g}$, is defined as

$\forall A \subseteq \Theta_h$, $m^{\uparrow g}(A^{\uparrow g}) = m(A)$, and $m^{\uparrow g}(B) = 0$ for all other $B \subseteq \Theta_g$.

$Bel^{\uparrow g}$ (*the extension of Bel to g*) is, as usual, characterized by $m^{\uparrow g}$.

References

Bartlett, M.S. (1952). The statistical significance of odd bits of information. *Biometrika* **39**, 228-237.

Gärdenfors, P. (1988). *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. The MIT Press, Cambridge, Massachussetts.

Good, I.J. (1956). The surprise index for the multivariate normal distribution. *Annals of Mathematical Statistics* **27**, 1130-1135.

Hsia, Y.-T. (1991a). Belief and surprise - a belief-function formulation. In *Proceedings of the Seventh Conference on Uncertainty in AI*, Los Angeles, California, 165-173.

Hsia, Y.-T. (1991b). Characterizing belief with minimum commitment. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Sydney, 1184-1189.

Hsia, Y.-T. (1992). Two notions of uncertainty in beliefs and their general incomparability. Technical Report TR/IRIDIA/92-5.

Levi, I. (1984). *Decisions and revisions*. Cambridge University Press.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, California.

Shackle, G.L.S. (1949). *Expectations in Economics*. Cambridge University Press.

Shackle, G.L.S. (1961). *Decision, Order and Time in Human Affairs*. Cambridge University Press.

Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton University Press.

Smets, P. (1988). Belief functions. In *Non-Standard Logics for Automated Reasoning* (P. Smets, E. H. Mamdani, D. Dubois and H. Prade eds.). Academic Press, London.

Weaver, W. (1948). Probability, rarity, interest and surprise. *Scientific Monthly* **67**, 390-392.

Zadeh, L.A. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems* **1**, 3-28.

A Continuous Belief Function Model for Evidential Reasoning

Chua-Chin Wang and Hon-Son Don
Department of Electrical Engineering
State University of New York
Stony Brook, NY 11794
U.S.A.

Abstract

A unified framework for evidential reasoning is proposed, in which the belief function of a piece of evidence is represented as a probability density function which can be in a continuous or discrete form. A vector form of mutual dependency relationship of the evidence is considered and a dependency propagation theorem is proved. An interpretation of belief conjunction from a geometrical view is proposed. This method can resolve the conflict resulting from either the mutual dependency among two pieces of evidence or the structural dependency in an inference network due to the evidence combination order. Belief conjunction, belief combination, belief propagation procedures, and AND/OR operations of an inference network based on the proposed framework are all presented, followed by some examples demonstrating the advantages of this method over the conventional methods.

1 Introduction

Evidential reasoning, which has been an essential part of many computational systems, is the task of assessing a certain hypothesis when certain pieces of evidence are given. The hypothesis is assessed by inferring its *belief value* from the belief values of different pieces of evidence. The belief value can be taken from a certain belief region, e.g., a unit interval $[0, 1]$, which can be discrete or continuous, or from a set of linguistic quantifiers, e.g., [*very unlikely, unlikely, likely, very likely*], etc. Regardless of the type of representation, the belief value of an evidence indicates the *belief strength* of that evidence. In describing the relationship between different pieces of evidence, *dependency* has been employed to describe the degree of truth of one piece of evidence implied by a second piece of evidence. If a hypothesis is supported by many pieces of evidence, then the combined belief strength of the hypothesis is not only dominated by the belief value of the individual evidence but also affected by the mutual dependencies among these evidence.

Because of the obscure and inexact nature of information, each piece of evidence is associated with some *uncertainty*. Reasoning with uncertainties in an inference network [3] includes three types of uncertainty aggregations: belief conjunction, belief combination, and belief propagation. There are three major frameworks of evidential reasoning in the literature, i.e., the Dempster-Shafer theory of evidence, the fuzzy set theory, and the Bayesian probability theory [7], [12]. The advantages and disadvantages of these three frameworks have been discussed in [1], [2], [14], [15], and [16]. The ability of Shafer's belief function to manage uncertainty of information in a rule-based system has attracted lots of attention [4], [5], [7], [9], [10], [13], [16], [19]. However, even with its strong popularity, the Shafer's model has drawbacks. Shafer's belief function model uses numerical values in the interval $[0, 1]$ to represent the degree of belief of information, which can also be interpreted as an index of inexactness of that information. The non-robustness of this model has been discussed by Hau [5], Zaheh [19], and Wang [17], [18]. Hau also has illustrated an example which shows the counter-intuitive result of Demspter's rule. Secondly, the basic probability assignment (BPA) of a belief function is in a form of discrete type function which can not always provide a precise description of an evidence for all the situations. It is often not appropriate to assign a discrete basic probability assignment over $[0, 1]$ by thresholding the interval into several regions, since the thresholds themselves can not describe the exact nature of a piece of evidence. The possible quantization problem caused by thresholding a continuous region for the weight of evidence has been discussed by Cheng [1], who provided an example showing that a quantized version of an associative belief combination may not necessarily be an associative one. The continuous form of a belief function, which is a more general representation, is needed to approximately express the vagueness of an evidence. The merit of a continuous form corresponds to the introduction of the membership function of fuzzy sets theory. Several other belief function approaches are included in [4] and [8], which have focused on handling the belief combination problem by using Dempster's rule to deal with belief propagation. However, these approaches didn't provide a formal proof of their respective methods. Although Hau [5] proposed an interpolative method to employ a factor to indicate

the degree of dependency between the independent case and maximally dependent case. It turns out to be incomplete, because the dependency could be bilateral instead of unilateral. In a rule-based intelligent system, the inconsistency of evidential reasoning resulting from the mutual dependency among different pieces of evidence and the structural dependency caused by the improper arrangement of an inference network has not been fully solved.

Although lots of efforts have been spent on belief combination, the uncertainty management of continuous belief function and the resolution of conflict due to the dependency of evidence are still not fully solved. The following is a proposed method which focuses on achieving conflict resolution of belief combination resulting from mutual dependency of evidence in an inference network. In addition, this method can also handle the information aggregation based upon the continuous belief functions. The nature of a belief function associated with an evidence is a probabilistic function, which could be discrete or continuous. We will discuss the belief conjunction first, since the belief combination and the belief propagation are both based on the belief conjunction.

2 Framework of The Continuous Belief Function Model

2.1 Representation of Evidence and Rule

The first step in the simulation of human reasoning with uncertainty is to find a proper way to represent that uncertainty and then build up the inference procedure. In the belief function introduced by Shafer [12] and Hau [5], two parameters, i.e., lower bound and upper bound, are employed to indicate *credibility* and *plausibility*. For the sake of clarity, the *belief function* is borrowed to represent the probability density function associated with a piece of evidence in the following text, and the belief function proposed by Shafer, [11], [12], [13] is named as *Shafer's belief function*. But, as discussed earlier, the probability assignment strategy for Shafer's belief function has its inherent flaw. For example, if a piece of evidence A is to emphasize that the closer it is to the truth, the stronger it is, then that evidence can be conveniently modeled by a continuous function, which is a probability density function,

$$Bel_A(\theta) = k \cdot \theta, \tag{1}$$

where θ is in the interval $[0, 1]$ indicating the degree of truth of the evidence, and k is a constant. We can hardly find any significant thresholds to quantize the associated belief function into a discrete form which can be handled by either Dempster-Shafer theory [12] or Hau's modified Demsper's rule [5]. Therefore, a more general representation of evidence is needed to represent such kind of uncertainty. In the following, we present our representation of the evidence and the inference rules.

Definition 1 : A piece of evidence in a rule-based system is represented by a subset A of the frame of discernment Θ, and a belief function associated with A is represented by a probability density function $p_A(\theta)$, where θ is a variable indicating the degree of truth for the evidence. $\overline{A}$ denotes the complement of A. 1 is used to denote the truth of the evidence and 0 is used to denote the falsity of the evidence. θ is a numerical value in the interval $[0, 1]$.

Note that the belief function is a probability density function in nature, the following property must hold,

$$\int_0^1 p_A(\theta)\, d\theta \;=\; 1. \tag{2}$$

This type of belief function can be transformed into a Shafer's belief function by assigning two bounds to the interval. For instance, if the above continuous belief function, Eqn. (1), is going to be transformed into the conventional belief function by choosing two thresholds in the belief region $[0, 1]$ and compute the respective area of each part so that the numerical values of the credibility and the plausibility of this belief function are obtained. If two thresholds, say $\frac{1}{3}$ and $\frac{2}{3}$, are chosen and k is 2 derived from Eqn. (2), the following results are obtained,

$$Cr_A = \int_{\frac{2}{3}}^1 2\theta d\theta = \frac{5}{9}, \quad Pl_A = \int_{\frac{1}{3}}^1 2\theta d\theta = \frac{8}{9}$$

On the other hand, given a Shafer's belief function by BPA method, e.g., a belief function with credibility $\frac{5}{9}$ and plausibility $\frac{8}{9}$, it can not precisely express the characteristics of the linear continuous belief function shown by Eqn. (1).

Definition 2 : A rule R in a rule-based system conveying uncertainty is represented as

$$R : E \longrightarrow H \qquad \text{with } p_{E \to H}(\theta)$$

where E is an evidence, H is a hypothesis implied by E, and $p_{E \to H}(\theta)$ is a probability density function to describe the degree of the truth of the rule. E is called *antecedent*, and H is called *consequent* of rule R.

In the above definition, the rule $R : E \longrightarrow H$ is interpreted as logic implication. In Section 1, we mentioned the inconsistency of evidential reasoning resulting from the mutual dependency among different pieces of evidence and the structural dependency caused by the various arrangements of an inference network. One example showing such inconsistency will be given in Section 2.3. The following definition of degree of *mutual dependency* is given to describe the relationship between two pieces of evidence.

Definition 3 : The degree of one evidence A depending on another evidence B is represented by ρ_{AB}, where by $0 \leq \rho_{AB} \leq 1$.

Note that ρ_{AB} is not necessarily equal to ρ_{BA}, which means a dependency relationship is a directed link. If ρ_{AB} equals 1, then it indicates A is totally dependent on B; if ρ_{AB} is 0, then A won't be affected by B at all. In the following, $p_{A \Rightarrow B}$ is used to denote the belief function of evidence A depending on evidence B, and ρ_{AB} is used to denote the coefficient of A depending on B.

2.2 Belief Conjunction

By definition, belief conjunction refers to the deduction of the belief associated with $(A \cap B)$ from the belief associated with evidence A and B, respectively. That is, given two frames of discernment Θ_A and Θ_B, a compatibility relation between Θ_A and Θ_B is the Cartesian product of them, which is represented as

$$\Theta_A \times \Theta_B \longrightarrow \Theta_{A \cap B} \tag{3}$$

There are three possible dependency relationships between two pieces of evidence A and B, which are

(i) A and B are independent;

(ii) A depends on B; and

(iii) B depends on A.

The relationship between two pieces of evidence can not be explicitly expressed by only one of the above relationships due to the vagueness and incompleteness of the evidence. In the following, we will develop a general formulation to cope with such a situation. Let

$$Conj(A, B) = A \cap B$$

and

$$\begin{aligned} p_{A\cap B}(\theta) &= \rho_I \cdot p_{indep}(\theta) + \rho_{AB} \cdot p_{A\Rightarrow B}(\theta) \\ &\quad + \rho_{BA} \cdot p_{B\Rightarrow A}(\theta) \\ &= \begin{bmatrix} \rho_I & \rho_{AB} & \rho_{BA} \end{bmatrix} \cdot \begin{bmatrix} p_{indep}(\theta) \\ p_{A\Rightarrow B}(\theta) \\ p_{B\Rightarrow A}(\theta) \end{bmatrix} \\ &= \vec{\alpha} \cdot \vec{P} \end{aligned} \tag{4}$$

The ρ's are dependency coefficients, which represent the degree of dependencies and $\rho_I + \rho_{AB} + \rho_{BA} = 1$. The $\vec{\alpha}$ is called *dependency vector*, and the $\vec{P}$ is called the *belief function vector*. There are three terms in (2), representing three different types of belief conjunction. Their meanings and how they can be computed are discussed in the following.

Case 1 : Independent

Referring to Fig. 1, since the two pieces of evidence A and B are independent, we assume that they can be located, respectively, on the two axes μ and ν of the Cartesian coordinates. The probability density function of the $A \times B$ is the multiplication of the probability density functions associated with individual evidence. That is,

$$p_{indep}(\mu, \nu) = p_A(\mu) \cdot p_B(\nu) \tag{5}$$

The above expression is a 2-D function forming a 2-D surface which should be reduced to 1-D form for the belief function of the conjunction result. Since these two pieces of evidence are independent with each other, they are considered to be equally important to the desired belief conjunction, $Conj_{indep}$, i.e., their contribution to the final result is the same. Therefore, for a ω, all of the density products $p_A(\mu) \cdot p_B(\nu)$, where $\mu + \nu = \omega$, should be attributed to $p_{indep}(\omega)$. The line $\mu = \nu$ is called *conjunction line*, and the resulted $p_{indep}(\omega)$ can be considered to reside on this line by the following calculations,

$$p_{indep}(\omega) = p_{indep}(\mu, \nu) \mid_{\omega=\mu+\nu=}$$

$$\begin{cases} \int_0^\omega p_A(\mu) \cdot p_B(\omega - \mu) d\mu & = \int_0^\omega p_B(\nu) \cdot p_A(\omega - \nu) d\nu \\ & \text{if } 0 \le \omega \le 1 \\ \int_{\omega-1}^1 p_A(\mu) \cdot p_B(\omega - \mu) d\mu & = \int_{\omega-1}^1 p_B(\nu) \cdot p_A(\omega - \nu) d\nu \\ & \text{if } 1 \le \omega \le 2 \end{cases} \tag{6}$$

where $\omega = \mu + \nu$. Obviously, the range of ω is $[0, 2]$. Also note that $\mu + \nu = \omega$ is a line perpendicular to the line $\mu = \nu$. The following propability conservation property can be easily proved.

Lemma 1 :

$$\int_0^2 p_{indep}(\omega) d\omega = 1 \tag{7}$$

However, we are aware of the assumed range of a belief function is $[0, 1]$. Therefore, normalization of the conjuncted belief function is necessary. This can be done by the following equation.

$$p_{indep}(\theta) = 2 p_{indep}(\omega) \mid_{\omega=2\theta} \tag{8}$$

Case 2 : Totally Dependent

This case includes two cases which are either A is totally dependent on B or B on A. Here we only discuss the former case, because the other is the same. Referring to Fig. 1, if A is totally dependent on B, then the resulted conjunction belief function should be the same as the belief function of B. This indicates that the $Conj_{A\Rightarrow B}(A, B)$ is exactly the B, that is, the conjunction line is rotated to overlap the ν-axis and the range of ω, which is the variable of the $Conj_{A\Rightarrow B}(A, B)$, is $[0, 1]$. On the other hand, in the case of B depending on A, we will have the conjunction line overlapping the μ-axis, and the $Conj_{B\Rightarrow A}(A, B)$ is exactly the A. Therefore, we have the following

$$p_{A\Rightarrow B}(\theta) = p_B(\theta) \tag{9}$$

$$p_{B\Rightarrow A}(\theta) = p_A(\theta) \tag{10}$$

The above Case 1 and 2 represent the extreme cases. The general case will lie in between these extreme cases and can be computed as an interpolation of these extreme cases according to the mutual dependency coefficients ρ_I, ρ_{AB} and ρ_{BA}. The result has been given in Eqn. (4). The following conservation result can be easily proved.

Lemma 2 :

$$\int_0^1 p_{Conj(A,B)}(\theta) d\theta = 1$$

2.3 Belief Combination

Belief combination refers to the belief conjunction of several pieces of evidence supporting the same goal hypothesis. Hau discussed in [5] a fact that the basic probability assignment method will introduce a conflict $(A \cap \overline{A})$ in belief combination because of the conjunction of evidence supporting the same hypothesis with different degree of belief. Therefore, a conflict resolution approach is needed to achieve the consistency of belief combination. By the theory of previous subsection, if

both A and B are supporting hypothesis C, then the probability density function shown on the conjunction line is the belief function of C. Therefore, equations (4) to (10) can also be applied to the combination of the two pieces of evidence, except that the two pieces of evidence must support the same hypothesis. The deficiency and nonrobustness of Dempster's rule have been detailedly discussed in Hau's work [5]. However, both Hau and Shafer ignored the vagueness of the dependency relationship between the two pieces of evidence. The mutual dependency relationship of pieces of evidence always introduces significant conflict in the reasoning result of an inference network. This conflict is shown in Fig. 2 and 3. Referring to Fig. 2, if there are three pieces of evidence, M, N, and K, supporting a hypothesis L, then in a sequential rule-based system, two of them have to be combined first, and then the result is combined with the third evidence. These two cases are shown in Fig. 3.

Example 1. Considering the two cases in Fig. 3, which have different structures. Assume

$$\begin{aligned} &Cr(M) = 0.98, \quad Pl(M) = 0.99, \quad \rho_{NK} = \rho_{KN} = 0.5, \\ &Cr(N) = 0.01, \quad Pl(N) = 0.02, \quad \rho_{NM} = 0.1, \\ &Cr(K) = 0.01, \quad Pl(K) = 0.99, \quad \rho_{KM} = 0.9. \end{aligned}$$

The inconsistent results of these two cases by Hau's method are tabulated in Table 1.

	Cr_L	Θ_L	$1 - Pl_L$
Case 1	0.006903	0.012857	0.980239
Case 2	0.003064	0.033357	0.963579

Table 1: The results of Hau's approach applied to cases of Fig. 3.

When compared with human's reasoning process, this kind of inconsistency is unimaginable. From the assumption, evidence M is the strongest one to support the hypothesis L, the other two pieces of evidence N and K are less important than M. According to Table 1, the resulted credibility of case 1 is more than twice of that of case 2. On the contrary, the plausibility of case 1 is only half of that of case 2. These results indicate that Hau's method is easily subject to the combination order of the evidence, which is not consistent with the intuition of human reasoning. To a human, if these three pieces of evidence are given, a belief function associated with L should be dominated by M, since the relative dependency ratios of N and K indicate their less influence on L, and because K has a stronger dependency on M than N, M should have the dominant impact on L. Therefore, we conclude that the results given by Hau's method are not consistent with human reasoning.

The reason why the conflict appears in Example 1 is the mutual dependency relationship will propagate through the inference network by current belief combination and then influence the following belief combination. Therefore, let's consider if the evidence is arranged in a latticed-structure inference network, as shown in Fig. 4, two pieces of evidence E_1 and E_2 are combined with each other so as to form an *intermediate evidence* E_3, and then E_3 and E_4 are combined to support the hypothesis H. *Intermediate evidence* means one piece of evidence synthesized by other evidence, and has all the properties that one piece of real evidence has. What have been given are the mutual relationships among E_1, E_2, and E_4. Therefore, before E_3 E_4 are combined, the relationship between E_3 and E_4 has to be determined. The following information is assumed to be known before the belief combination is performed.

$$\begin{aligned} \rho_{12} + \rho_{21} + \rho_{I_{12}} &= 1 \\ \rho_{14} + \rho_{41} + \rho_{I_{14}} &= 1 \\ \rho_{24} + \rho_{42} + \rho_{I_{42}} &= 1 \end{aligned}$$

where all ρ's are known, and ρ_I's denote the independency coefficients. The following result can be proved.

Interpolative Dependency Propagation Theorem : For the inference network shown in Fig. 4 and the above given information, the mutual relationships of E_3 and E_4 can be determined by the following equations

$$\begin{aligned} \rho_{34} &= \frac{\rho_{12}}{\rho_{12}+\rho_{21}} \cdot \rho_{24} + \frac{\rho_{21}}{\rho_{12}+\rho_{21}} \cdot \rho_{14} \\ \rho_{43} &= \frac{\rho_{12}}{\rho_{12}+\rho_{21}} \cdot \rho_{42} + \frac{\rho_{21}}{\rho_{12}+\rho_{21}} \cdot \rho_{41} \\ \rho_{I_{34}} &= \frac{\rho_{12}}{\rho_{12}+\rho_{21}} \cdot \rho_{I_{42}} + \frac{\rho_{21}}{\rho_{12}+\rho_{21}} \cdot \rho_{I_{14}} \end{aligned} \tag{11}$$

Proof : Using the vector notation of Section 2.2, let dependency vectors $\vec{\alpha}_{14}, \vec{\alpha}_{24}, \vec{\alpha}_{34}$ denotes the mutual dependency relationships between E_1 and E_4, E_2 and E_4, E_3 and E_4, respectively,

$$\begin{aligned} \vec{\alpha}_{14} &= [\rho_{I_{14}} \quad \rho_{14} \quad \rho_{41}] \\ \vec{\alpha}_{24} &= [\rho_{I_{24}} \quad \rho_{24} \quad \rho_{42}] \\ \vec{\alpha}_{34} &= [\rho_{I_{34}} \quad \rho_{34} \quad \rho_{43}] \end{aligned}$$

Since E_3 is the combination result of E_1 and E_2, the following equation holds,

$$\vec{\alpha}_{34} = \frac{\rho_{12}}{\rho_{12}+\rho_{21}} \cdot \vec{\alpha}_{24} + \frac{\rho_{21}}{\rho_{12}+\rho_{21}} \cdot \vec{\alpha}_{14}$$

which proves the theorem.

Example 2. Given the same data as in Example 1, we use our method, i.e., equations (4) to (11), to process each combination in the two cases in Fig. 3. The final results are listed in Table 2.

	Cr_L	Θ_L	$1 - Pl_L$
Case 1	0.106148	0.879853	0.013998
Case 2	0.105937	0.868451	0.025612

Table 2: The results of proposed model applied to either case of Fig. 3.

In Table 2, it is obvious that both cases have almost the same credibility Cr_L and plausibility $Cr_L + \Theta_L$, which means our model will not be seriously influenced by the order of the belief combination compared with the result derived by Hau's modified Dempster's rule, and meets the intuition of human reasoning. The conflict appearing in Example 1 has been resolved by our

model. Here we have introduced a vector form for mutual dependency shown in (4) and the above dependency propagation theorem, which turns out to be superior to the conventional scalar form for the dependency between different pieces of evidence in resolving conflict caused by the dependency problem.

2.4 Belief Propagation

Belief propagation refers to the aggregation of the uncertainty associated with the evidence or fact to fire a rule and the uncertainty of the rule itself so as to deduce the uncertainty of the goal hypothesis of the rule. Referring to Fig. 5, given a rule $R : P \longrightarrow Q$ and an evidence P, then we are interested in exploring the belief function of the conclusion Q supported by the evidence P, which is defined as the belief propagation result of the evidence P and the rule R. Because it is impossible to obtain the exact belief function of the consequent Q from the given evidence and rule, what we can expect is an assessment of the *maximum bound* and *minimum bound* of the belief function associated with Q. If an evidence A is covered by another evidence B, i.e., the information of A is contained in that of B, then we denote their relationship by $A \subseteq B$. Using this notation, the relationship among the maximum bound Q_{max}, the minimum bound Q_{min}, and Q can be expressed as

$$Q_{min} \subseteq Q \subseteq Q_{max}.$$

Let $p_{P \to Q}(\theta)$ be the belief function associated with the rule R and $p_P(\theta)$ be the belief function associated with the evidence P. The conjunction of the rule and the evidence is

$$(P \to Q) \cap P = (\overline{P} \cup Q) \cap P = (Q \cap P)$$

The conjunction result $(Q \cap P)$ means the consequent Q holds when it is supported by the antecedent P. Therefore, the belief function of this conjunction provides the minimum bound of the belief function of Q, i.e., $Q_{min} = (Q \cap P)$. This result meets the definition which we have explored for the belief propagation. Therefore, by applying the conjunction procedure of Section 2.2 to a piece of evidence and a rule, we will get the result of belief propagation. In other words, the belief function obtained by the conjunction of the belief functions of the antecedent and the rule is actually the minimum bound of the belief function of the consequent.

However, we are also interested in assessing the maximum bound of the belief function of the consequent Q. Referring to [4] and [5], the smallest range of $\overline{Q}$ is $(P \cap \overline{Q})$ which can also be derived from basic logic operations. Hence, we conclude that

$$Q_{max} = \overline{(P \cap \overline{Q})} = P \to Q,$$

which implies the maximum bound of the belief function of Q is exactly the same as the belief function of the given rule despite what the belief function of P is. In other words, if the maximum bound is employed as the belief propagation result, then the uncertainty of the antecedent will not have any impact on the belief propagation. This is not true at all. Note that many researchers have proposed various approaches to recover and assess the belief function of the consequent, and provided many explanations to their methods, e.g., [4] and [5]. However, we are only interested in the impact provided by the antecedent to the consequent which shows the degree of the antecedent supporting the consequent. Henceforth, we adopt only the minimum bound of the belief function of the consequent in a belief propagation procedure to assess the uncertainty aggregation of belief propagation.

In propositional logic, the logic implication, $A \to C$, can be synthesized by the conjunction of other logic implications, for example, the rules $A \to B$ and $B \to C$. It is easy to derive the following "*chaining syllogism*".

Lemma 3 : Given two rules

$$R_1 : A \to B, \quad R_2 : B \to C,$$

with the belief functions $p_{R_1}(\theta)$ and $p_{R_2}(\theta)$, respectively. The belief function $p_{R_3}(\theta)$ associated with the new rule, $R_3 : A \to C$, is the conjunction of the two belief functions, $p_{R_1}(\theta)$ and $p_{R_2}(\theta)$.

2.5 AND/OR operations

In an inference network, the function of each node is either AND or OR operation. In order to analyze the uncertainty aggregation of an inference network, it is necessary to consider these two operations for belief function. In Section 2.2, we mentioned that the conjunction of two pieces of evidence are deemed as the AND operation of the two pieces of evidence, which is described as $Conj(A, B) = A \cap B$. Therefore, all of the theory of the belief conjunction given in Section 2.2 can be applied to the AND operation.

An OR operation for two pieces of evidence can be defined as $Union(A, B) = A \cup B$. From basic logic theory, we know

$$A \cup B = A + B - A \cap B,$$

which means the following equation holds,

$$p_{Union(A,B)}(\theta) = p_A(\theta) + p_B(\theta) - p_{Conj(A,B)}(\theta) \quad (12)$$

where $p_{Conj(A,B)}(\theta)$ can be derived according to the model introduced in Section 2.2.

Since an inference network can be viewed as an AND/OR graph, if all of the evidence, rules, and the mutual dependency relationships are given, the belief function of the hypothesis can be derived by the proposed model. The procedure is first to perform the operations of low-level nodes and then gradually propagate towards the root where the hypothesis resides.

3 Simulation Examples

In this section, we use some examples to illustrate the theory and procedures of the proposed evidential reasoning model presented in the previous sections.

Example 3. There are two pieces of evidence A and B supporting a hypothesis C. Assume the property of A supporting C is the more A is true, then the more C

is also true. However, if B has an opposite property to that of A, and B is partially dependent upon A with an degree 0.4, then what is the belief that A and B support C?

We know that the properties of two evidence A and B given in the above can not be easily modeled by the *basic probability assignment* (BPA) method to be processed by Shafer-Dempter's rule. However, it can be handled by the proposed model. First, we can assign a belief function to the evidence A based on its property,

$$p_A(\theta) = 2\theta$$

On the other hand, we can assign a belief function to the evidence B,

$$p_B(\theta) = 2 - 2\theta$$

Following the procedure discussed in Sections 2.2, and 2.3, first, three dependency coefficients must be computed. Since there is no information to indicate there is a possibility that A depends upon B either partially or totally, we can assume that

$$\rho_{B\Rightarrow A} = 0.4, \quad \rho_{A\Rightarrow B} = 0.0, \quad \rho_{I_{AB}} = 0.6$$

Secondly, we calculate the $p_{indep}(\theta)$, which can be deduced by equations (3), (4), (5), and (6).

$$p_{indep}(\theta) = \begin{cases} 16\theta^2 - \frac{32}{3}\theta^3, & 0 \le \theta \le \frac{1}{2} \\ \frac{16}{3} - 16\theta^2 + \frac{32}{3}\theta^3, & \frac{1}{2} \le \theta \le 1 \end{cases}$$

Then, since A and B support the same hypothesis, we can apply (2) to get the overall belief function of hypothesis C, which is

$$\begin{aligned} p_C(\theta) &= 0.4p_A(\theta) + 0.6p_{indep}(\theta) \\ &= \begin{cases} \frac{4}{5}\theta + \frac{48}{5}\theta^2 - \frac{32}{5}\theta^3, & 0 \le \theta \le \frac{1}{2} \\ \frac{16}{5} + \frac{4}{5}\theta - \frac{48}{5}\theta^2 + \frac{32}{5}\theta^3, & \frac{1}{2} \le \theta \le 1 \end{cases} \end{aligned}$$

The graphs of belief functions, $p_A(\theta)$, $p_B(\theta)$, and $p_{indep}(\theta)$ are shown, respectively, in Fig. 6, 7, and 8. The graph of the belief function $p_C(\theta)$ associated with the hypothesis C is plotted in Fig. 9. Referring to Fig. 6-9, the curve of the belief function $p_C(\theta)$ is dominated by the independent case of belief conjunction of the two given evidence, which is in Fig. 8, and the right hand side of the graph indicates the perturbation provided by the B depending on A case, which is in Fig. 6. Therefore, the probability density distribution of the belief function p_C meets what the intuition of human reasoning expects.

Example 4. Given an inference rule R and evidence A and B, which are shown as an inference network in Fig. 10, we are going to assess the belief function of the consequent supported by the two given evidence. Suppose that the node combining A and B is an OR node, which will constitute an intermediate evidence. The rule is

$$R: if \ (A \ OR \ B) \ then \ C,$$

where the belief functions of evidence A and B, and the mutual dependency relationships between the evidence are the same as those given in Example 3, and the belief function of the rule R is

$$p_R(\theta) = 3 \cdot \theta^2,$$

which is shown in Fig. 11. We also assume that the rule R is independent of its antecedent. In the following, we will derive the belief function of the consequent C by our model.

By the result of the previous example, we can draw a conclusion that the belief function of the intermediate evidence D, which is an OR node, is

$$\begin{aligned} p_D(\theta) &= p_A(\theta) + p_B(\theta) - p_{A\cap B}(\theta) \\ &= \begin{cases} 2 - \frac{4}{5}\theta - \frac{48}{5}\theta^2 + \frac{32}{5}\theta^3, & 0 \le \theta \le \frac{1}{2} \\ \frac{-6}{5} - \frac{4}{5}\theta + \frac{48}{5}\theta^2 - \frac{32}{5}\theta^3, & \frac{1}{2} \le \theta \le 1 \end{cases} \end{aligned}$$

which is shown in Fig. 12. Since the rule R is independent on any antecedent, the dependency relationship between D and R must be

$$\rho_{DR} = 0, \quad \rho_{RD} = 0, \quad \rho_{indep} = 1$$

Here, for the sake of simplicity, we rewrite the belief function p_D as the following

$$\begin{aligned} p_D(\theta) &= p_A(\theta) + p_B(\theta) - p_{A\cap B}(\theta) \\ &= \begin{cases} F_1(\theta), & 0 \le \theta \le \frac{1}{2} \\ F_2(\theta), & \frac{1}{2} \le \theta \le 1 \end{cases} \end{aligned}$$

Then, according the theory of belief conjunction given in Section 2.2, we can get the following equations,

$$\begin{aligned} &p_C(\theta) \\ &= \begin{cases} \frac{1}{25}(1024\theta^6 - 1536\theta^5 - 160\theta^4 + 800\theta^3), \\ \qquad\qquad 0 \le \theta \le \frac{1}{4} \\ \frac{1}{400}(9600\theta^2 - 1728\theta + 112) - \frac{1}{25}(1024\theta^6 \\ \quad -1536\theta^5 + 160\theta^4 + 480\theta^3), \frac{1}{4} \le \theta \le \frac{1}{2} \\ \frac{1}{400}(9600\theta^2 - 1728\theta + 112) - \frac{1}{25}(1024\theta^6 \\ \quad -1536\theta^5 - 160\theta^4 - 1760\theta^3 + 4800\theta^2 \\ \quad -2360\theta + 280), \qquad \frac{1}{2} \le \theta \le \frac{3}{4} \\ \frac{1}{25}(1024\theta^6 - 1536\theta^5 + 160\theta^4 - 2080\theta^3 + \\ \quad 4800\theta^2 - 2824\theta + 456), \qquad \frac{3}{4} \le \theta \le 1 \end{cases} \end{aligned}$$

which is shown in Fig. 13.

The overall propagation result has been derived and the property of propability conservation holds. The capability of the proposed model in managing the uncertainty aggregation of complicated continuous belief functions of evidence and inference rules in an inference network is illustrated in above Example 4. If the conventional BPA approach is adopted in this example, no matter what thresholds are chosen to quantize the belief region, the fiedility of the original belief function will be seriously distorted or even totally lost. In turn, regardless of what kind of belief combination and belief propagtion method is employed, it will mislead to an unfaithful resulted belief function of the final hypothesis.

4 Conclusion

The conflict caused by the Dempster's rule [12] or Hau's approach [5] can not be satisfactorily resolved. In contrast, the proposed framework solve this conflict resolution problem. The result of belief propagation must also depend on the mutual dependency relationship of the antecedent of the rule and the rule itself, which has

long been ignored in the literature, in addition to the interpretations of the rule. The proposed model also embodies the inference in a lattice-structured inference network. Since an inference network can be treated as an AND/OR graph, the operation of any individual node (i.e., AND or OR) can reach the final result by applying the proposed model. If the node is AND, then it is considered to be a *Conj* operation in the proposed model, while an OR node, then *Union* operation.

The new model offers several advantages over prior attempts. First, the conflict in the dependencies among the evidence in an inference network has been solved. Secondly, the discrete belief functions and the arbitrary continuous belief functions can be processed, which has not been researched to date. A continuous belief function is extremely advantageous because it can represent the vagueness of a human concept more accurately than conventional means (i.e., BPA). Thirdly, the problem dependency propagation of an intermediate evidence has been resolved. Finally, the conflict of belief propagation caused by the mutual dependency relationship of the antecedent of the rule and the rule itself has also been solved. This newly proposed strategy is intuitively closer to the intuitive reasoning process than is Dempster's consensus seeking strategy [11] [12] or Hau's compromise seeking strategy [5], thus making this new model more successful and appealing than earlier attempts.

References

[1] Y. Cheng and R. L. Kashyap, "A study of associative evidential Reasoning," *IEEE Trans. Pattern Anal. Machine Intell.*, vol.11, no.6, June 1989.

[2] P. R. Cohen, *Heuristic reasoning about uncertainty: an artificial intelligence approach,* Morgan Kaufmann Publishers, Inc., Los Altos, California, 1985.

[3] R. O. Duda, P. E. Hart, and N. Nilsson, "Subjective Baysian methods for rule-based inference systems," *Proc. AFIPS,* 1976.

[4] M. L. Ginsberg, "Nonmonotonic reasoning using Dempster's rule," *Proc. National Conf. Artificial Intell.* 1984, pp. 126-129.

[5] H. Y. Hau and R. L. Kashyap, "Belief combination and propagation in a lattice-structured inference network," *IEEE Trans. Systems Man Cybernet,* vol. 20, no.1, Jan./Feb. 1990.

[6] R. A. Hummel and M. S. Landy, "A statiscal viewpoint on the theory of evidence," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PAMI-10, no. 2, Mar. 1988.

[7] J. Ihara, "Extension of conditional probability and measures of belief and disbelif in hypothesis based on uncertain evidence," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PAMI-9, no. 4, Jul. 1987

[8] M. Ishizuka, K. S. Fu, and J. T. P. Yao, "Inference procedure with uncertainty for problem reduction method," *Information Sci.,* vol. 28, pp. 179-206, 1983.

[9] K. B. Laskey, and P. E. Lehner, "Assumptions, beliefs and probability," Artificial Intelligence, 41 (1989/90) 65-77.

[10] J. Pearl, "Fusion, propagation, and structuring in belief networks," Artificial Intelligence, 29 (1986) 241-288.

[11] G. Shafer and R. Logan, "Implementing Dempster's rule for hierachical evidence," *Artificial Intell.,* vol. 33, 1987.

[12] G. Shafer and J. Pearl, *Readings in uncertain reasoning,* Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[13] P. P. Shenoy and G. Shafer, "Propagating belief functions with local computation," *IEEE Expert,* pp. 43-51, Fall 1986.

[14] H. Tahani and J. M. Keller, "Information fusion in computer vision using the fuzzy integral," *IEEE Trans. Systems Man Cybernet,* vol. 20, no.3, May/June 1990.

[15] L. E. Widman, K. A. Loparo and N. R. Nielsen, *Artificial intelligence, simulation, and modeling,* John Wiley & Sons Inc., 1989.

[16] J. Yen, "Generalizing the Dempster-Shafer theory to fuzzy sets," *IEEE Trans. Systems Man Cybernet,* vol. 20, no.3, May/June 1990.

[17] C.-C. Wang, and H.-S. Don, "A new approach to evidential reasoning by a potential model," *Proc. ISMM Int. Conf. Computer Applications in Design, Simulation and Analysis,* Las Vegas, Mar 1991.

[18] C.-C. Wang, and H.-S. Don, "A geometrical approach to evidential reasoning," *IEEE Int. Conf. Systems, Man, and Cybernetics,* Charlottesville, Virginia, Oct. 1991.

[19] L. A. Zadeh, "A simple view of the Dempster-Shafer theory of evidence and its implication for the rule of combination," *Fuzzy Set Syst.,* vol. 11, 1983

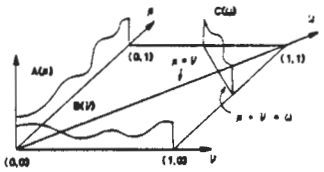


Fig. 1 The independent case of the 2-D model.

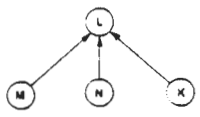
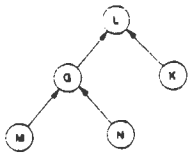


Fig. 2 Combination of three evidences to support a hypothesis.

Case 1:



Case 2:

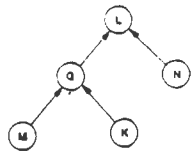


Fig. 3 Two different structures of combination of three evidences.

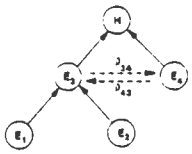


Fig. 4 An inference network for demonstrating the mutual dependency relationship between E_2 and E_4.

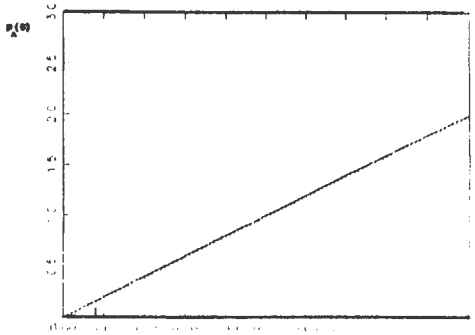


Fig. 5 Belief function of the evidence A in Example 3.

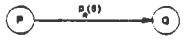


Fig. 6 Representation of an inference rule R.

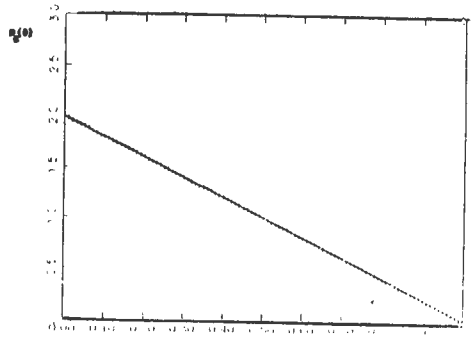


Fig. 7 Belief function of the evidence B in Example 3.

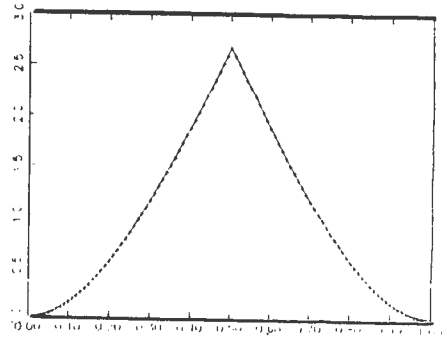


Fig. 8 Belief function of the independent case for A and B in Example 3.

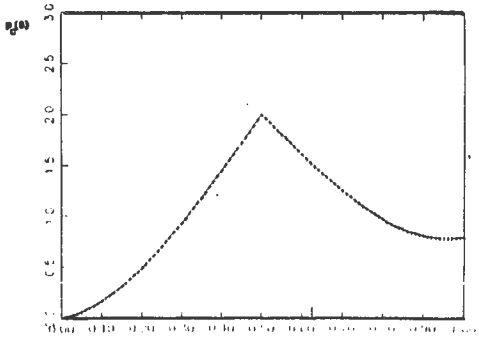


Fig. 9 Belief function of the hypothesis C in Example 3.

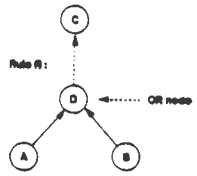


Fig. 10 The inference network of Example 4.

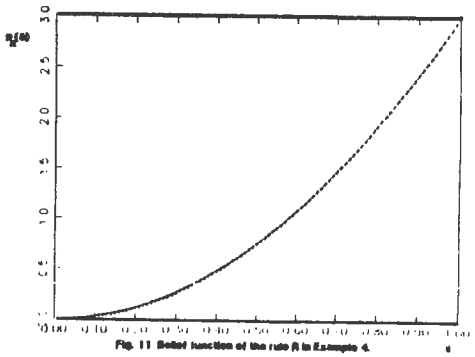


Fig. 11 Belief function of the rule R in Example 4.

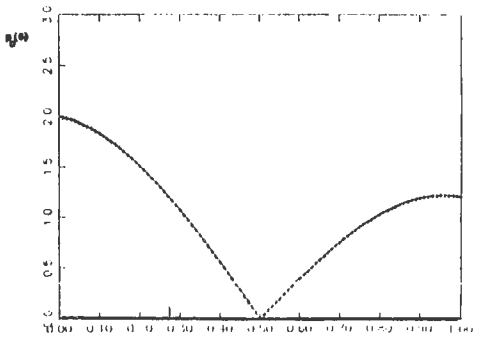


Fig. 12 Belief function of the intermediate evidence D in Example 4.

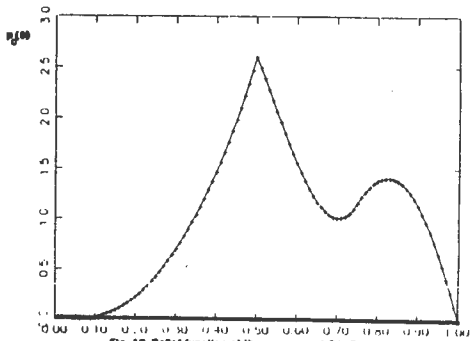


Fig. 13 Belief function of the consequent C in Example 4.

An Efficient Approach to Probabilistic Inference in Belief Nets *

Zhaoyu Li and **Bruce D'Ambrosio**
303 Dearborn Hall
Department of Computer Science
Oregon State University
Corvallis, OR 97331-3202
U. S. A.
zhaoyu@cs.orst.edu
dambrosi@cs.orst.edu

Abstract

A number of exact algorithms have been developed to perform probabilistic inference in belief nets in recent years. In general these algorithms have been developed from a graph-theoretic perspective. In this paper we consider probabilistic inference in a belief net from the combinatorial optimization point of view. Efficient probabilistic inference in a belief net can be seen as the problem of finding an optimal factoring given a set of probability distributions to be combined. From this viewpoint, previously developed algorithms are just different factoring strategies. We begin by defining the optimal factoring problem. We then present evidence for the utility of this perspective in the form of a very simple heuristic algorithm for factoring in multiply-connected networks. We show experimental evidence that this simple greedy heuristic is more efficient than previously developed exact probabilistic inference algorithms.

1 Introduction

A number of exact algorithms have been developed to perform probabilistic inference with belief nets in recent years. In general these algorithms have been developed from a graph-theoretic perspective[1]. Among the algorithms that have been developed are the propagation algorithms based on the original directed graph, such as the poly-tree propagation algorithm [Pearl, 1988; Pearl, 1986; Peot and Shachter, 1991], on a related directed graph, such as Symbolic Probabilistic Inference [D'Ambrosio, 1989; D'Ambrosio and Shachter, 1990] and Shachter's reduction algorithms [Shachter, 1986; Shachter, 1989; Shachter, 1988], or on a related undirected graph, such as Pearl's clustering and conditioning algorithms [Pearl, 1988] and Lauritzen and Spegielhalter's clustering algorithm [Lauritzen and Spiegelhalter, 1988].

The time complexity of probabilistic inference in a belief net is exponential in the number of nodes in the graph in the worst case. However, practical belief nets are usually only sparsely connected and we can take the advantage of conditional independence represented by sparseness of the graph to reduce the complexity of computation; that is, we may perform some non-numerical computation to reduce numeric computation. So, probabilistic inference in a belief net can generally be considered as consisting of two components: numeric computation of the requested distributions and non-numeric computation or symbolic reasoning. For the above exact algorithms, the non-numeric computation of each algorithm is heuristic and is performed in polynomial time in the number of nodes of a belief net; but the time cost of numeric computation is exponential with respect to the maximum number of variables of two distributions to be combined in a query [Li, 1990]. Therefore, the key point for speeding up a probabilistic inference algorithm is to reduce its numeric computation time. There are two ways to reduce the numeric computation time. One is to reduce the maximum dimensionality in the computation by optimal arrangement of the computations to be performed. The second is to reduce the size of the computation by pre-computation or the re-structuring of the graph off-line. Pre-computation is not always applicable. In this paper we will restrict our attention to finding more efficient factorings. Unfortunately, finding an optimal factoring is a hard problem.

Previous algorithms for inference in belief nets can be viewed as various heuristic strategies for factoring a product of a set of distributions. However, these strategies do not explicitly address the fundamental combinatorial optimization problem, and so may miss opportunities for optimization. We begin by defining a combinatorial optimization problem: *Optimal Factoring*. Then, we present a heuristic factoring algorithm. Finally, we show experimental results of the algorithm on randomly generated belief nets and compare these results with those obtained by using the symbolic probabilistic inference (SPI) algorithm [D'Ambrosio, 1989], one of the most efficient exact algorithms [Li, 1990].

The remainder of the paper is organized as follows.

*The authors gratefully acknowledge the support by NSF 91-00530.

[1] Shenoy's work [Shenoy, 1991] is very similar in spirit to ours. He studies inference under uncertainty from an algebraic perspective. However, he emphasizes mathematical foundations, whereas we emphasize algorithm development.

Section 2 introduces a combinatorial optimization problem - optimal factoring. Section 3 describes a heuristic factoring strategy we developed for multiply connected belief nets. Section 4 shows experimental tests of the algorithm. Section 5 discusses the test results. Finally, section 6 discusses conclusions of the research.

2 Optimal factoring

An optimal factoring problem (OFP) with n expressions can be considered as a combinatorial optimization problem. Without loss of generality we assume that the domain size of each variable is 2. The problem can be described as follows.

Definition 1 (FP) *Given*

1. *a set of m variables V,*
2. *a set of n subsets of V: $S = \{S_{\{1\}}, S_{\{2\}}, ..., S_{\{n\}}\}$, and*
3. *$Q \subseteq V$ is a set of target variables*

define operations:

1. *combination of two subsets S_I and S_J:*

$$S_{I\cup J} = S_I \cup S_J - \{v : v \notin S_K \text{ for } K \cap I = \phi, \; K \cap J = \phi, \text{ and } v \notin Q\},$$
$$I, J \subseteq \{1, 2, ..., n\}, \; I \cap J = \phi;$$

2. *the cost function of combining the two subsets:*

$$\mu(S_{\{i\}}) = 0 \text{ for } 1 \leq i \leq n, \text{ and}$$
$$\mu(S_{I\cup J}) = \mu(S_I) + \mu(S_J) + 2^{|S_I \cup S_J|}.$$

$\mu(S_I)$ is not unique if $|I| > 2$. In general, it depends on how we combine the subsets. We indicate these alternative combinations by subscripting μ. $\mu_\alpha(S_I) = \mu$ shows the cost of combining result of S_I with respect to a specific tree structured combination of I, labeled α. We call this combination a factoring.

Definition 2 (OFP) *An* **optimal factoring** *problem is to find a factoring α such that $\mu_\alpha(S_{\{1,2,...,n\}})$ is minimal.*

In above definitions, Q is a set of target variables after combining all subsets of S together; the set $\{v\}$ in the formula $S_{I\cup J}$ is the variables which do not appear in the remaining subsets of S after combination of S_I with S_J. $\mu_\alpha(S_{I\cup J})$ is the total cost of combining all set S_i ($i \in I, J$) in a given factoring order, and is determined by the cardinalities of sets to be combined and affected by the size of $\{v\}$ in previous combinations. If the domain size of each variable is not limited to 2, the value $2^{|S_I \cup S_J|}$ in above formula should be replaced with the multiplication of domain size of the variables in $S_I \cup S_J$. All possible factorings are equivalent to the results of permuting the n variables and then putting parentheses in all possible ways in the permutation to form all $S_{\{1,2,...,n\}}$ [2]. It should be clear that determining the lowest cost way to combine distributions in a belief net is an OFP. The initial set of subsets is the set of sets each containing one variable (or node) and its parents. The reason that we do not restrict the appearance of variables in each subset S_i is that some intermediate combination results or distributions still meet the definitions.

OFP generally is a hard problem. We suspect that OFP is an NP-hard problem, although do not yet have a proof for this conjecture. Despite this, some restricted instances of OFP may have polynomial time algorithms. For example, given a domain of variables, if each pair of sets S_i and S_j is disjoint and the set Q is the union of all the sets S_i; then the optimal ordering of $\alpha(S_{\{i_1,...,i_n\}})$ could be obtained in linear time. Our specific interest is in the application of OFP to probabilistic inference.

One direct use of OFP is to find an optimal evaluation tree for computing queries in a belief net. Given a belief net with m nodes and some observations in it, a query involves identification of a subset n of the nodes relevant to the query, and computation of the conformal product [Shachter *et al.*, 1990] of marginal and conditional probabilities of the n nodes. The n nodes with their relations can be mapped to the symbols in the definition of OFP: the n nodes with their immediate antecedent nodes are mapped to the n subsets of m variables; the queried nodes correspond to the variables in the subset Q; $S_{I\cup J}$ denotes the intermediate result of conformal product of distributions I and J; and μ gives the number of multiplications needed for the computation. Finding an optimal factoring minimizes the number of multiplications needed for this computation[3].

From the OFP point of view, we can see that previously developed exact probabilistic inference algorithms are just different factoring strategies. However, these strategies didn't consider probabilistic inference as a factoring problem. We have proved [Li and D'Ambrosio, 1991] that there exists an optimal factoring algorithm which runs in polynomial time in the number of nodes for a poly-tree. However, finding an optimal factoring for an arbitrary belief net is a hard problem. In the next section we will present a heuristic algorithm for factoring an arbitrary belief net and show experimentally that this simple algorithm performs extremely well by testing it on randomly generated belief nets and comparing its results with those from SPI.

3 A heuristic factoring Algorithm

We now present an efficient heuristic algorithm, called set-factoring, we have developed for finding good factorings for probability computation. In a belief net with nodes $\{x_1, x_2, .., x_n\}$ connected by arcs, the general form of query is $P(X_J|X_K, X_E)$, where X_J is a set of nodes being queried, X_K is a set of conditioning nodes and X_E is a set of observed nodes. $P(X_J|X_K, X_E)$ can be computed from $P(X_J, X_K|X_E)$. For simplicity, we will only consider the case $P(X_J|X_E)$ in the algorithm. This ignores several potential simplifications noted in [Shachter *et al.*, 1990], but simplifies the presentation.

Given a query $P(X_J|X_E)$ in a belief net, often only a sub-set of the nodes is involved in the probability computation. The involved nodes can be chosen from the

[2] The parentheses show the preference of combination.

[3] For proofs of the properties of probabilistic models which enable this algebraic approach, see [Shachter *et al.*, 1990].

original belief net by an algorithm which runs in time linear in the number of nodes and arcs in the belief net [Geiger *et al.*, 1989]. Once we have obtained the nodes needed for the query, we have all the factors to be combined. In accordance with the definition 1, we have n sub-sets of n nodes and set Q. We use the following algorithm to combine these factors.

1. Construct a *factor set* A which contains all factors to be chosen for the next combination. Each factor in set A consists of a set of nodes. Initialize a *combination candidate set* B empty.
2. Add any pairwise combination of factors of the factor set A to B if the combination is not in set B, except the combination of two factors in which each factor is a marginal node and they have no common child; and compute the $u = (x \cup y)$ and sum(u) of each pair. Where x and y are factors in the set A, sum(u) is the number of nodes in u which can be summed over when the probability computation corresponding to the two factors is carried out.
3. Choose elements from set B such that $C = \{u | u : minimum_B(|u| - sum(u))\}$, here $|u|$ is the size of u excluding observed nodes. If $|C| = 1$, x and y are the factors for the next combination; otherwise, choose elements from C such that $D = \{u | u : maximum_C(|x|+|y|),\ x, y \in u\}$. If $|D| = 1$, x and y are the terms for the next multiplication, otherwise, choose any one of D.
4. Generate a new factor by combining the candidate pair chosen from above steps and modify the factor set A by deleting two factors of the candidate pair from the factor set and putting the new factor in the set.
5. Delete any pair of set B which has non-empty intersection with the candidate pair.
6. Repeat step 2 to 5 until only one element is left in the factor set A which is the final combination.

Following is an example to illustrate the algorithm by using the net shown in figure 1. Suppose that we want to compute the query p(4) for the belief net and assume that there are 2 possible values of each variable. The nodes relevant to the query are {1, 2, 3, 4}. We use the set-factoring algorithm to get their combination.

Loop1: factor set A is {1,2,3,4}; The set B is {(1,2) (1,3) (1,4) (2,3) (2,4) (3,4)} after step 2; the current combination is (1,2), i.e. $p(2|1) * p(1)$ after step 3 (there was more than one candidate in this step, we chose one arbitrarily); the set A is {(1,2), 3, 4} after step 4; and the set B is {(3,4)} after step 5.

Loop2: factor set A is {(1,2),3,4}; The set B is {((1,2),3) ((1,2),4)) (3,4)} after step 2; the current combination is ((1,2),3) after step 3; the set A is {(1,2,3), 4} after step 4; and the set B is empty after step 5.

Loop3: factor set A is {(1,2,3),4}; The set B is {((1,2,3),4)} after step 2; the current combination is ((1,2,3),4) after step 3; the set A is {(1,2,3,4)} after step 4; and the set B is empty after step 5. The factoring result is

$$p(4) = \sum_{2,3}(p(4|2,3)(\sum_{1}(p(3|1)(p(2|1)p(1))))).$$

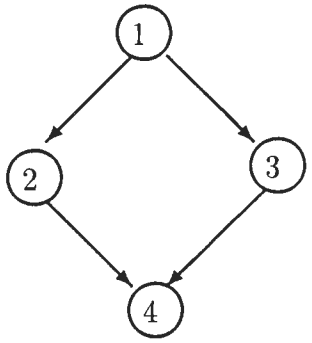


Figure 1: A simple belief net.

There are several things that should be noticed in the algorithm. First, queried nodes should not be deleted from any terms in the expression, and if a node is a queried node and it has no parents then the node will be combined after all other nodes are combined. Second, we assume that the number of the values of all nodes is same in the algorithm. If the numbers of values of the nodes in a belief net are different, we can consider the product of the number of values of all nodes related in each step instead of the number of nodes. Third, a caching strategy can be used in the algorithm. A caching table is generated before any query. Before combining any two factors, we check the caching table to see if there is a cached result for the combination. If there is a cached result, we can use the cached result at a cost of 0 instead of doing the real probability computation. If there is no such a cached result, then real computation will be carried out. This caching strategy will save some computation time.

The heuristic strategy in the algorithm can be explained as follows: In step 2, $(x \cup y)$ shows the number of multiplications needed for combining the pair x and y[4]. The elements in the set B are the candidates for the next combination. We don't consider pairs consisting of two unrelated marginal nodes if they don't have common children since a combination of the two marginal nodes will usually increase dimensionality. In step 3, we choose the pairs which have the lowest result dimensionality as candidates since the best result of the current combination may need less multiplications than those of the other combinations for subsequent combinations. The effect of summation is considered here; it always decreases the dimensionality of the result. If more than one candidate is generated here, we choose the maximum $(|x| + |y|)$ in step 4 as a criterion because this choice maximizes the number of variables being summed over. Usually, it is better to sum over variables as early as possible. Steps 4 and 5 are just preparations for the next loop.

The time complexity of the algorithm is dominated by the number of nodes related to the current query. Step

[4]The number of multiplications should be $2^{|x \cup y|}$.

1 is linear in the number of nodes. In step 2, there are $n*(n-1)/2$ pairs to be computed for the set B at the first loop, and (n- k) new pairs are added in the set at the end of the kth loop. There is a total of $(n*(n-1)/2)+\sum_k(n-k)=(n^2-3n+3)$ pairs are computed. For each pair, the union operation is O(m), here m is the maximum size of x; and sum(u) can be computed at the same time as computing $(x \cup y)$. So the time complexity in step 2 is $O(n^3)$ at most. The time cost of step 3 is linear in the number of pairs left in the set B and set C respectively, it is at most $O(n^2)$ including (n-1) loops needed for the two steps. Modifying the factor set in step 4 is linear in the number of factors, it has at most n elements. Deleting some elements from the set B in step 5 is linear in the number elements in the set. The time complexity is $O(n^2)$ in step 4 and $O(n^3)$ in step 5 including (n-1) loops for the algorithm. Therefore, the time complexity of the algorithm is $O(n^3)$ in the number of nodes.

4 Experimental tests

Time complexity of some currently used exact probabilistic inference algorithms, conditioning, clustering, reduction and SPI, have been analyzed, and their efficiency has been experimentally tested [Li, 1990] with the implementation of IDEAL system [Srinivas and Breese, 1989] for conditioning, clustering and reduction algorithms and the implementation of SPI [D'Ambrosio, 1989]. Since SPI had better performance in that study than the others, we just experimentally compare set-factoring with SPI in this section.

Three sets of test cases were generated for time complexity experiments. We used J. Suermondt's random net generator to generate all test cases. This generator starts with a fully connected belief net of size n, and removes arcs selected at random until the number of the remaining arcs is equal to a selected value. In each test case, we randomly[5], (ranging from 1 to the number of nodes in the belief net), determined the number of observations to be inserted in that test case; then we randomly chose each observation from all unobserved variables in the belief net and finally we chose at random a set of variables as queries from remaining variables after each observation. The number of multiplications needed for each test case was recorded.

The first set of test cases is randomly generated with from 1.0 to 3.0 arcs per node and 8 to 13 nodes. The reason for choosing a set of small belief nets for testing is because we want to compare the results of set-factoring with those of an optimal algorithm, which is limited to run small belief nets because of time complexity[6]. Table 1 shows the characteristics of the 10 test cases and the computational results of different algorithms measured in the number of multiplications: **nodes** is the number of nodes in each belief net; **arcs/node** is the average arcs per node; **obs** is the number of observations inserted in the belief net; and **query** is the total number of queries. The last three columns give the results of the generalized SPI [D'Ambrosio and Shachter, 1990], the set-factoring algorithm and the optimal algorithm respectively. From the table we see that the set-factoring has a better factoring result than the generalized SPI but is not optimal in two test cases.

The second set of test cases is tree-structured belief nets. They are randomly generated with from 10 to 30 nodes. Table 2 shows the 10 belief nets and the test results. The columns 2 to 4 show the number of nodes, the number of observations and the number of queries for each test case. The columns 5 to 7 show the test results for each algorithm as in table 1. From the table we see that the set-factoring has an optimal result for each tree structured belief net. The generalized SPI did not get optimal results for some test cases.

The third set of test cases is that used in testing SPI and Generalized SPI [D'Ambrosio, 1989; D'Ambrosio and Shachter, 1990; Shachter *et al.*, 1990]. They are randomly generated from 1.0 to 5.0 arcs per node, and 10 to 30 nodes. In table 3, **node** shows the number of nodes and **arc** shows the number of arcs in each belief net; the columns **obs** and **query** give the the number of observations and total queries in each test case respectively; and the rest of columns shows the number of multiplications for each test case. A new version of SPI is used for comparison. **SPI(cache)** and **set-fact(cache)** show the results with intermediate result caching for both algorithms[7].

From the above experimental results we see that the factoring strategy of set-factoring has better factoring results than those of SPI in every case, particularly when a belief net is large. The number of multiplications in set-factoring is about half of those in SPI on average. Set-factoring is more consistent with respect to tasks and different kinds belief nets. As shown in table 3, set-factoring is better than SPI with caching for a large belief net, taking net 16 as an example. Since dimension in a factor will become large after some combinations, any bad combination order will cause many more multiplications than a good combination does.

The time complexity of factoring for set-factoring and the time complexity of symbolic reasoning for SPI are only slightly different. In set-factoring, the time complexity is $O(n^3)$ in the number of nodes concerned in the current query at most; in SPI the time complexity is $O(n^3)$ at most in the number of nodes of the belief net. There should be no big difference. The time cost for symbolic reasoning in both algorithms is trivial compared to probability computation.

5 Discussion

While these results are preliminary, they seem a strong indication that the set-factoring algorithm is able to find better factoring for many problems, particularly in finding optimal factoring for all tree test cases. Also the set-factoring algorithm can be used as a suitable ana-

[5] Unless noted otherwise, all random selections are from uniform distributions over the indicated range.

[6] The optimal algorithm is a dynamic programming algorithm with exponential cost.

[7] * denotes that corresponding algorithm is too slow to run the test case.

net	nodes	arcs/node	obs	query	G.SPI	set-factoring	opt-alg
1	12	2	3	7	287	52	52
2	11	2.5	3	7	328	196	196
3	9	2.5	4	12	301	252	252
4	11	2	4	4	58	26	26
5	9	2.2	1	3	140	102	102
6	8	2.6	2	4	200	194	186
7	13	1	3	7	109	38	38
8	13	2.5	3	8	2760	1818	1716
9	13	2.4	3	8	144	94	94
10	10	1.7	3	7	237	174	174

Table 1: Ten small test cases and the test results by algorithms: the generalized SPI, the set-factoring and the optimal algorithm.

net	nodes	obs	query	G.SPI	set-factoring	opt-alg
1	23	6	68	728	646	646
2	19	19	89	1881	630	630
3	28	1	4	36	36	36
4	22	16	104	2959	1246	1246
5	17	7	34	809	404	404
6	12	9	27	335	148	148
7	24	17	128	1469	68	68
8	25	1	10	222	178	178
9	24	5	58	1478	1010	1010
10	22	5	46	1427	642	642

Table 2: Tree structured test cases and test results by algorithms: the generalized SPI, the set-factoring and the optimal algorithm.

net	node	arc	obs	query	SPI	set-factoring	SPI(cache)	set-fct(cache)
1	23	28	10	13	164	98	140	60
2	13	62	7	6	832	718	368	310
3	13	61	10	4	62	44	32	28
4	18	85	10	8	624	558	422	418
5	16	54	8	9	2,370	1,512	866	898
6	17	34	8	9	2,616	890	1,176	502
7	23	60	10	12	37,514	5,272	10,078	2,978
8	10	15	5	5	286	182	222	92
9	27	35	13	14	1,122	644	800	244
10	12	26	5	7	780	386	452	194
11	23	87	10	12	183,296	73,804	65,216	26,540
12	11	36	5	6	1,896	1,126	668	598
13	14	15	7	6	454	228	264	92
14	16	40	8	8	8,416	3,112	2,204	1,940
15	19	76	9	10	81,696	23,590	13,380	10,462
16	29	131	1	28	*	6,569,756	16,146,192	3,196,900
17	29	90	14	14	1,489,040	143,334	254,292	73,146
18	16	35	9	6	2,480	898	816	450
19	15	53	7	8	15,986	4,168	3,068	1,896
20	26	101	13	13	717,552	124,734	113,248	63,834
21	28	34	14	13	2,052	847	1,384	330

Table 3: The experimental results of 21 test cases between SPI and set-factoring.

lytical tool for evaluating other probabilistic inference algorithms. The most important conclusion from the experimental results is that OFP is a useful way of efficiently solving probabilistic inference problems in a belief net. From the OFP point of view, not only can we get a better algorithm than previously those developed, but also the algorithm is easy to understand and implement.

The main idea behind the set-factoring algorithm is, at each step, to find a pair with the best combination result. We tried the strategy of finding the pair with minimum multiplication as a candidate for combination; the results are not as good as those obtained by set-factoring. The set-factoring algorithm only considers information locally for choosing each pair, so it can be implemented efficiently. It is this characteristic that prevents the algorithm from guaranteeing an optimal result for some multiply connected belief nets, because optimal results are related to all nodes concerned. It also tells us why the algorithm is good in tree structured belief nets: the factoring information for the tree is locally determined. Due to the locality of its heuristic strategy, set-factoring can work as a local factoring strategy in other probabilistic inference algorithms.

Since the last several combinations in factoring usually have large dimensionality, combinations of the last few factors are critical in getting nearly optimal result for some belief nets. Considering this, we combined the set-factoring and the optimal algorithm together to get a new algorithm in which we used set-factoring to generate a partial result first and then used the optimal algorithm to complete the last several combinations. Since the optimal algorithm can run efficiently for about 8 factors, the combined algorithm should run efficiently as well. The results of the combined algorithm are better than the set-factoring algorithm, particularly for large belief nets[8]. This lead us to think of another factoring strategy of using the optimal algorithm. That is, if a belief net can be divided into several connected parts, we might use optimal algorithm within each part and then among all parts. We have not tested this idea yet.

The test result of the net 3 in table 3 for set-factoring (without caching) is optimal for each query, but both algorithms with caching give better results for the same queries. This indicates that a best probabilistic inference algorithm may not only depend on an optimal factoring strategy, but also depend on a good caching method for some tasks and some belief nets. There is a trade off between using a good factoring strategy and using an effective caching method in an inference algorithm, since a good factoring strategy flexible across many belief nets and tasks may be hard to combine with any caching method.

We have also studied the opportunities for parallelism in belief net inference. Set-factoring has shown good factoring results for parallel probability computation [D'Ambrosio *et al.*, 1992].

[8]Take the net 16 in table 3 as an example, the number of multiplications needed by the combined algorithm is about 75% of those by set-factoring.

6 Conclusions

In this paper we have presented a combinatorial optimization problem, optimal factoring. We have proposed that efficient probabilistic inference in a belief net can be considered as an optimal factoring problem. We believe that it is a proper way to study the problem. From this point of view, finding an efficient exact probabilistic inference algorithm is the problem of finding an optimal factoring algorithm. Unfortunately, finding an optimal factoring in general is a hard problem. Currently developed algorithms rely on structural properties of the graph to guide factoring. However, it is not clear this is the most direct way to find efficient factorings. We presented a heuristic factoring algorithm for multiply connected nets which makes no reference to graphical structure and yet outperforms current graph based algorithms.

References

[Srinivas and Breese, 1989] Sampath Srinivas and Jack Breese. IDEAL: Inference Diagram Evaluation and Analysis in Lisp, Documentation and Users Guide. Rockwell International Science Center, Palo Alto Laboratory, May 1989

[D'Ambrosio, 1989] Bruce D'Ambrosio. Symbolic Probabilistic Inference in Belief Networks. OSU technical report. December 1989.

[D'Ambrosio *et al.*, 1992] Bruce D'Ambrosio, Tony Fountain and Zhaoyu Li. Parallelizing Probabilistic Inference - Some Early Explorations. Submitted to the Eighth Conference on Uncertainty in Artificial Intelligence.

[D'Ambrosio and Shachter, 1990] Bruce D'Ambrosio and R. Shachter. Factoring Heuristics for Generalized SPI. OSU technical report. March 1990.

[Geiger *et al.*, 1989] D. Geiger, T. Verma and J. Pearl. d-separation: From Theorems to Algorithms. *Proceedings of the fifth Workshop on Uncertainty in Artificial Intelligence*, pages 118-125. Windsor, Ontario, 1989.

[Lauritzen and Spiegelhalter, 1988] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Royal Statistical Society*, pages 157-224. January 1988.

[Li, 1990] Zhaoyu Li. Complexity of Probabilistic Inference in Belief Nets - an Experimental Study. MS thesis, OSU. November 1990.

[Li and D'Ambrosio, 1991] Zhaoyu Li and Bruce D'Ambrosio. Probabilistic Inference in Belief Nets - A Combinatorial Optimization Problem. OSU, Oct, 1991.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, San Mateo, 1988.

[Pearl, 1986] J. Pearl. Fusion, Propagation, and Structuring in Belief Networks. *Artificial Intelligence*, 29(3): 241-288, 1986.

[Pearl, 1990] J. Pearl. Reasoning with Belief Functions: An Analysis of Compatibility. *International Journal of Approximate Reasoning*, vol 4, pages, 263-289. September/November 1990.

[Peot and Shachter, 1991] Mark A. Peot and Ross D. Shachter, Fusion and Propagation with Multiple Observations in Belief Networks. *Artificial Intelligence*, 48: 299-318, 1991.

[Shachter, 1986] R. D. Shachter. Evaluating Influence Diagrams. *Operation Research*, vol 34, pages 871-882, November 1986.

[Shachter, 1989] R. D. Shachter. Evidence Absorption and Propagation Through Evidence Reversal. *Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence*, pages 303-310, 1989.

[Shachter, 1988] R. D. Shachter. Probabilistic Inference and Influence diagrams. *Operations Research*, Vol. 36, pages 589-604, July-August, 1988.

[Shachter *et al.*, 1990] R. D. Shachter, Stig K. Anderson and Kim L.Poh. Directed Reduction Algorithms and Decomposable graphs. *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 237-244, 1990.

[Shachter *et al.*, 1990] R. D. Shachter, B. D'Ambrosio, and B. D. Favero. Symbolic Probabilistic Inference in Belief Networks. *Proceedings, Eighth National Conference on AI*, pages 126-131, 1990.

[Shenoy, 1991] Prakash
Shenoy. Independence In Valuation-based Systems, School of Business, University of Kansas, Nov. 1991.

Definite Integral Information

Scott D. Goodwin
Dept. of Computer Science
University of Regina
Regina, Saskatchewan,
Canada, S4S 0A2
(306) 585-4700
goodwin@cs.uregina.ca

Eric Neufeld
Dept. of Computational Science
University of Saskatchewan
Saskatoon, Saskatchewan,
Canada, S7N 0W0
(306) 966-4887
eric@USask.ca

André Trudel
Jodrey School of Computer Science
Acadia University
Wolfville, Nova Scotia,
Canada, B0P 1X0
(902) 542-2201
trudel@AcadiaU.ca

Abstract

Given vague temporal information about an interval, we wish to draw reasonable inferences about interior points and subintervals. Within the deductive framework of probability, such queries are sufficiently underconstrained as to result in degenerate answers. We show that when vague temporal information is represented in the form of *definite integral information*, we can nonmonotonically infer answers to such queries that are both more interesting and useful. The inductive inference is achieved by identifying temporal intervals with reference classes and choosing the most specific. We also consider the effect of other knowledge—for instance, the knowledge that temporal information is true throughout some subinterval.

1 Introduction

We study a previously undefined yet important class of temporal interval based information we call *definite integral information*. An example of a problem involving such information is:

> Over the year, it rains 40% of the time. During winter (December 21–March 20), it rains 75% of the time. Over the summer (June 21–September 20), it rains 20% of the time. What percentage of rainfall occurs during December? What is the chance of rain at midnight on December 24th?

To obtain useful answers to these questions, it is necessary to go beyond what deductively follows from probability theory.

The non-deductive reasoning we wish to do with definite integral information requires nonmonotonic "jumping to conclusions" about subintervals and points based on interval information. In the absence of "other information," we assume that every point has an equal chance of having some property. We usually do have "other information" and want to consider each point with respect to its *most specific reference class*, the narrowest class to which it belongs and for which we have "adequate" information [Reichenbach, 1949; Kyburg, 1983]. Our view of what constitutes the narrowest reference class differs from the commonly held view. We adopt recent ideas about inheritance of statistical information where information about a class of individuals has (the usual) *downward influence* on subclasses *and* has (a previously unconsidered) *upward influence* on superclasses [Goodwin, 1991]. This lets us deal with conflicting sources of statistical knowledge by averaging the influences (i.e., a sort of "maximum entropy weighted average").

Such inferences require reasonable inductive assumptions. Here we develop a probabilistic framework in which subinterval and point information is nonmonotonically inferred directly from definite integral information. This direct inference hinges on an assumption of *epistemological randomness*, that is, if according to our current knowledge a particular point is interchangeable with any other point in some interval for which we have definite integral information, then the probabilistic properties of the point can be reasonably assumed to correspond to statistical properties of the interval. In a similar fashion, particular subintervals can reasonably be assumed to inherit statistical properties from the interval. The problem of applying definite integral information to subintervals and points is related to the idea of *direct inference* in statistical reasoning [Pollock, 1984]. Direct inference is the application of statistical information about a reference class to a member of the class. When there are multiple applicable reference classes, we must choose between them—this is the *reference class problem* [Reichenbach, 1949; Kyburg, 1983]—here we adopt the reference class selection policy proposed in [Goodwin, 1991].

We also consider the effect of information such as continuity. If we know the percentage of rainfall during an interval, the additional knowledge that the rain fell continuously constrains the assumptions we can make about the distribution. In Section 5, we discuss how to reason in this case.

2 Definite integral information

The first order temporal logic we use is a variant of GCH [Trudel, 1990]. Point based information is represented with a function $\beta(t)$ which equals unity if β is true at time point t, and equals zero otherwise. For example, "walking at time 3" is represented as $walking(3) = 1$. Our unorthodox representation of point-based informa-

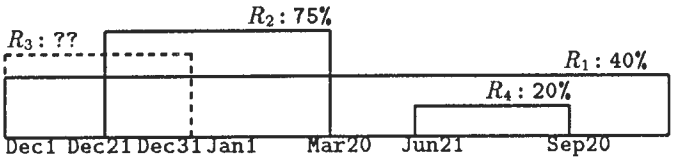


Figure 1: Rainfall Summary

tion facilitates the representation of interval based information. Since the function $\beta(t)$ is a 0-1 valued function, we can integrate it over an interval to get the duration of truth of β over the interval. For example, if John is walking continuously between times 0 and 7, then:

$$\forall t.\ 0 \leq t \leq 7 \rightarrow walking(t) = 1.$$

We integrate to get the length of time (in this case 7 time units) that John is walking over the interval [0,7]:

$$\int_0^7 walking(t)dt = 7. \quad (1)$$

The value of the integral need not equal the length of the interval. For example, walked for 6 hours during some 10 hour period is represented as:

$$\int_{t_0}^{t_0+10} walking(t)dt = 6. \quad (2)$$

Note that walking may not have occurred for 6 consecutive hours. It may have been interrupted many times.

Interval based information β is defined to be *definite integral information* if it can be represented in terms of a definite integral:

$$\int_{t_1}^{t_2} \beta(t)dt = \alpha$$

where $0 \leq \alpha \leq (t_2 - t_1)$. For example, formulas (1–2) describe definite integral information.

The rainfall example from Section 1 (summarized in Figure 1) is another example involving definite integral information. The GCH representation is given in Figure 2. Here $jan1.s$ and $jan1.e$ refer to the start and end of January 1st respectively.

3 Examples

Using the rainfall example of Figure 1, we present different examples of reasoning with definite integral information.

3.1 Interior point inferences

How likely is it to be raining at midnight on December 24th? That is, what is the chance that $raining(dec24.e) = 1$. Notice that $dec24.e$ falls within the following reference classes (refer to Figure 1):

$R_1 = (jan1.s, dec31.e]$,
$R_2 = (jan1.s, mar20.e] + (dec21.s, dec31.e]$,
$R_3 = (dec1.s, dec31.e]$.

$$\int_{jan1.s}^{dec31.e} raining(t)dt = 0.4 \times (dec31.e - jan1.s)$$
$$\int_{dec21.s}^{dec31.e} raining(t)dt = 0.75 \times (dec31.e - dec21.s)$$
$$\int_{jan1.s}^{mar20.e} raining(t)dt = 0.75 \times (mar20.e - jan1.s)$$
$$\int_{jun21.s}^{sep20.e} raining(t)dt = 0.2 \times (sep20.e - jun21.s)$$
$$jan1.s = 0 \ \wedge\ jan1.e = 1 \ \wedge \ldots \wedge\ dec31.e = 365$$

Figure 2: Rainfall Formulas

R_3 is not useful because we cannot deduce any useful statistics about it from the axioms of Figure 2 (and the axioms of probability). We also choose to ignore R_1 since R_2 is a narrower (more specific, more relevant) reference class. We base our inferences about $dec24.e$ on the reference class R_2. According to our current knowledge, all the points within R_2 are indistinguishable *with respect to rainfall* so we assume every point in R_2 has the same chance of rain and conclude a 75% chance of rain at time $dec24.e$.

3.2 Subinterval inferences

What percentage of rainfall occurs in December? The actual percentage of rainfall in December is:

$$\frac{\int_{dec1.s}^{dec31.e} raining(t)dt}{(dec31.e - dec1.s)}$$

where the value of the numerator is unknown. To compute the amount of rainfall in December we divide the month (region R_3 from Figure 1) into subregions $R_{3a} = (dec1.s, dec20.e]$ and $R_{3b} = (dec21.s, dec31.e]$.

The narrowest reference class that contains R_{3a} is

$$R_5 = (mar21.s, jun20.e] + (sep21.s, dec20.e]$$
$$= R_1 - R_2 - R_4.$$

The statistic for R_5 can be computed from the statistics for R_1, R_2, and R_4, and from the relative sizes of these intervals. We compute the *actual* percentage of rain P_5 over R_5 to be:

$$P_5 = \frac{0.4 \times |R_1| - 0.75 \times |R_2| - 0.2 \times |R_4|}{|R_5|}$$
$$= \frac{0.4 \times 365 - 0.75 \times [79 + 11] - 0.2 \times 92}{365 - [79 + 11] - 92}$$
$$\approx 33\%.$$

By assuming *every subset* of R_5 has the same *expected* percentage of rain, we conclude the expected percentage of rain over R_{3a} is P_5.

The narrowest reference class (for which we have or can compute the actual percentage of rainfall) that contains R_{3b} is R_2. By assuming *every subset* of R_2 has

the same *expected* percentage of rain, we conclude the expected percentage of rain over R_{3b} is 75%.

The *expected* percentage of rain over R_3 equals a *weighted* average based on R_{3a} and R_{3b}:

$$\begin{aligned} P_3 &= \frac{P_5 \times |R_{3a}| + 0.75 \times |R_{3b}|}{|R_3|} \\ &\approx \frac{0.33 \times 20 + 0.75 \times 11}{31} \\ &\approx 48\%. \end{aligned}$$

The answer to the original question is that it rains roughly half the time during December.

4 Direct inference rules

Let us now provide a formal characterization of the plausible inferences described in the previous section.

Definite integral information corresponds to a set of probabilistic constraints which determine a set of possible interpretations. In general, we can only deduce weak constraints on subintervals and points from definite integral information. For instance, from the rainfall example of Figure 2, the definite integral information that tells us that it rains 75% of the time in the winter, deductively implies that it rains between 50% and 100% of the time in the first half of winter. As for the individual points, we can only deduce that it may or may not be raining.

The plausible inferences we wish to make go beyond those deductively entailed by the definite integral information. They hinge on an assumption of epistemological randomness (or, roughly, a principle of indifference); that is, taking all we know into account, each possible interpretation is assumed to be interchangeable (i.e., equally expected). From this we can infer an expected value for the function β at a particular point. We can compute this value from the given definite integral information by first determining an interval of points, all of which have the same expected value for β, and using definite integral information about the interval to determine the expected value at the particular point. This interval of points having the same expected value for β is a *maximally specific reference interval.*

Definition 1 (Max. Specific Ref. Interval)
The (possibly non-convex) interval R is a *maximally specific reference interval* for $\beta(t_0)$ if

1. $t_0 \in R$, and
2. β has the same expected value at every point in R.

We do not need to know the expected value of β at every point in an interval to determine if it is maximally specific—we only need to know the expected values are the same. If the information we have concerning two points is identical (with respect to β), then the expected value of β at the points is the same. So all we need do is verify that we have the same information for every point in the interval.

Definition 2 (Interchangeable Points)
Suppose the only definite integral information our knowledge base contains concerning β is for the intervals: $R_1, R_2, \ldots, R_n$ (we refer to these as *explicit reference intervals* for β). We say two points t_1 and t_2 are *interchangeable* with respect to β, by which we mean that β has the same expected value at t_1 and t_2, if for every explicit reference interval R_i for β, we have $t_1 \in R_i$ iff $t_2 \in R_i$.

This says if the points fall within the same explicit reference intervals, then the expected values are (defined to be) the same.

Proposition 3 (Finding Max. Spec. Ref. Int.)
If I is the intersection of the explicit reference intervals for β that contain the point t_0, and U is the union of the explicit reference intervals for β that *do not* contain the point t_0, then a maximally specific reference interval for $\beta(t_0)$ is $R = I - U$.

Once we have identified a maximally specific reference interval for $\beta(t_0)$, we can relate the expected value of $\beta(t_0)$ to the definite integral information concerning the reference interval. To do this, we use the following property of mathematical expectation:

Proposition 4 (Expectation of Sums)
For any interval I

$$E\left(\int_I \beta(t)dt\right) = \int_I E(\beta(t))dt.$$

Interval I need not be convex. For example, if $I = (x, y) \cup (u, v)$ and (x, y) and (u, v) are disjoint, then $\int_I = \int_x^y + \int_u^v$.

Another useful property of mathematical expectation is that the expected value of a constant is the constant.

Proposition 5 (Properties of Expectations)
For any interval I, expressions X, Y, and constant c

$$\begin{aligned} E(c) &= c \\ E(cX) &= cE(X) \\ E(X + Y) &= E(X) + E(Y). \end{aligned}$$

We use Proposition 4 and the fact that the points of the reference interval have the same expected value for β to derive the following direct inference rule:

Proposition 6 (Direct Inference Rule 1)
If R is a maximally specific reference interval for $\beta(t_0)$ then

$$E(\beta(t_0)) = E\left(\frac{\int_R \beta(t)dt}{|R|}\right).$$

The expected value of β at t_0 is equal to the average value of β over the interval R. Note that this follows trivially from the fact that β has the same value at every point in R since R is a maximally specific reference interval.

In the rainfall example (Figures 1 and 2), we compute the expected value for *raining*(*dec*24.*e*) as follows.

First, we identify a maximally specific reference interval for *raining*(*dec24.e*) using Proposition 3. Here we have R_1 and R_2 as the explicit reference intervals for *raining* that contain *dec24.e* and R_4 is the only explicit reference interval not containing *dec24.e*. So $R = (R_1 \cap R_2) - R_4 = R_2$ is a maximally specific reference interval. The result $E(raining(dec24.e)) = 75\%$ now follows from Direct Inference Rule 1, our definite integral information about R_2, and Proposition 5:

$$
\begin{aligned}
& E(raining(dec24.e)) \\
&= E\left(\frac{\int_{R_2} raining(t)dt}{|R_2|}\right) && \text{Prop. 6} \\
&= \frac{E\left(\int_{R_2} raining(t)dt\right)}{|R_2|} && \text{Prop. 5} \\
&= \frac{E\left(0.75 \times |R_2|\right)}{|R_2|} \\
&= \frac{0.75 \times |R_2|}{|R_2|} && \text{Prop. 5} \\
&= 75\%
\end{aligned}
$$

Another example is the computation of the expected value of *raining*(*dec7.e*):

$$
\begin{aligned}
& E(raining(dec7.e)) \\
&= E\left(\frac{\int_{R_5} raining(t)dt}{|R_5|}\right) && \text{Prop. 6} \\
&= \frac{E\left(\int_{R_5} raining(t)dt\right)}{|R_5|} && \text{Prop. 5}
\end{aligned}
$$

where $R_5 = R_1 - (R_2 + R_4)$ is a maximally specific reference interval for *raining*(*dec7.e*) (by Proposition 3). We compute the numerator as follows:

$$
\begin{aligned}
& E\left(\int_{R_5} raining(t)dt\right) \\
&= E\left(\int_{R_1 - R_2 - R_4} raining(t)dt\right) \\
&= E\left(\int_{R_1} raining(t)dt\right. \\
&\quad - \int_{R_2} raining(t)dt \\
&\quad \left. - \int_{R_4} raining(t)dt\right) \\
&= E(0.4 \times 365 - 0.75 \times 90 - 0.2 \times 92) \\
&= 60.1.
\end{aligned}
$$

The value of the denominator is: $|R_5| = 365 - 90 - 92 = 183$. So therefore: $E(raining(dec7.e)) = 60.1/183 \approx 33\%$.

The following property is useful in relating subintervals to intervals:

Proposition 7 (DIRECT INFERENCE RULE 2)
If S is a (possibly non-convex) subinterval of a maximally specific reference interval R for β then

$$E\left(\frac{\int_S \beta(t)dt}{|S|}\right) = E\left(\frac{\int_R \beta(t)dt}{|R|}\right).$$

An example which uses Proposition 7 is the computation of the expected value of *raining* over December. As in Section 3.2, we divide December into two subintervals R_{3a} and R_{3b}. By Proposition 3, we identify $R_5 = R_1 - R_2 - R_4$ as a maximally specific reference interval containing R_{3a} and R_2 as a maximally specific reference interval containing R_{3b}. We then use Direct Inference Rule 2 to relate the subintervals to the reference intervals. The computation of the expected values from the definite integral information about the reference intervals is then straightforward (see Appendix A).

As a final example, suppose we know (see Figure 3):

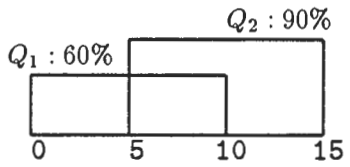


Figure 3: Overlapping reference intervals.

$$\int_0^{10} \beta(t)dt = 6 \ \wedge \ \int_5^{15} \beta(t)dt = 9.$$

Notice the explicit reference intervals $Q_1 = [0, 10]$ and $Q_2 = [5, 15]$ overlap. For $\beta(7)$, we have $Q_3 = Q_1 \cap Q_2 = [5, 10]$ is a maximally specific reference interval. We derive from the definite integral information that

$$\int_5^{10} \beta(t)dt \in [4, 5]$$

and then by Direct Inference Rule 1, we have that

$$E\left(\beta(7)\right) = E\left(\frac{\int_5^{10} \beta(t)dt}{5}\right) \in [\tfrac{4}{5}, 1].$$

5 Continuous definite integral information

If in addition to

$$\int_0^{10} \text{walking}(t)dt = 7,$$

we know that walking is uninterrupted (i.e., once the person starts walking, he walks for 7 consecutive time units), then this type of temporal information is called *continuous definite integral information*. Interval based information β is defined to be *continuous definite integral information* over some interval (t_1, t_2) if:

1. $\int_{t_1}^{t_2} \beta(t)dt = \alpha$ where $0 \leq \alpha \leq (t_2 - t_1)$, and
2. there exists a subinterval (t_3, t_4) of (t_1, t_2) such that $\int_{t_3}^{t_4} \beta(t)dt = t_4 - t_3 = \alpha$.

The direct inference rules from Section 4 are not applicable to continuous definite integral information. Instead, we use the formulas presented below.

5.1 Interior point inferences

Given that walking is continuous definite integral information over the interval (0,10) and that $\int_0^{10} \text{walking(t)}dt = 7$, what is the chance that walking is true at times 0, 1, or 5? Intuition suggests that walking is false at time 0 (there are an infinite number of subintervals of (0,10) of length 7 and only one of these subintervals contains the element 0) and true at time 5 (5 is an element of each subinterval of (0,10) of length 7).

Formally, suppose we have continuous definite integral information about β over $R = (R_a, R_b)$ where R is an interval of interchangeable points (see Definition 2) and $\int_R \beta(t)dt = L$. To compute the expected value of β at a point t_0 in R, we use the proportion of subintervals of length L in R that contain t_0:

$$E(\beta(t_0)) = \frac{\min(t_0, R_b - L) - \max(R_a, t_0 - L)}{R_b - L - R_a}. \quad (3)$$

The term $min(t_0, R_b - L)$ is the latest time and $max(R_a, t_0 - L)$ is the earliest time that a subinterval of length L that includes t_0 could start. The denominator represents all the possible start times.

Returning to the walking example, the probability of walking at $t_0 = 0$ is $(0 - 0)/3 = 0$. The probability at $t_0 = 5$ is $(3 - 0)/3 = 1$ and the probability at $t_0 = 1$ is $(1 - 0)/3 = 1/3$.

5.2 Subinterval inferences

Once again, given that walking is continuous definite integral information over the interval (0,10) and that $\int_0^{10} \text{walking(t)}dt = 7$, what proportion of points in $(0, 1)$ is walking true?

More generally, if (t_0, t_1) is a subinterval of $R = (R_a, R_b)$, where R is defined as in Section 5.1 and letting $K = R_b - L - R_a$, then:

$$E\left(\int_{t_0}^{t_1} \beta(t)dt\right) = \int_{t_0}^{t_1} E(\beta(t))dt$$

Using equation 3, we substitute the integrand and obtain

$$\begin{aligned}
&K^{-1} \int_{t_0}^{t_1} \min(R_b - L, t) - \max(R_a, t - L)dt \\
&= K^{-1} \left(\int_{t_0}^{t_1} \min(R_b - L, t)dt \right. \\
&\qquad \left. - \int_{t_0}^{t_1} \max(R_a, t - L)dt \right) \\
&= K^{-1} \left[\left(\int_{\min(t_0, R_b - L)}^{\min(t_1, R_b - L)} t dt \right. \right. \\
&\qquad \left. + \int_{\max(R_b - L, t_0)}^{\max(R_b - L, t_1)} (R_b - L)dt \right) \\
&\qquad - \left(\int_{\min(t_0, R_a + L)}^{\min(t_1, R_a + L)} R_a dt \right. \\
&\qquad \left. \left. + \int_{\max(R_a + L, t_0)}^{\max(R_a + L, t_1)} (t - L)dt \right) \right].
\end{aligned}$$

Returning to the walking example, we expect walking to be true at all points in (5, 6), at 1/6 of the points in (0, 1) and at 0.958333 of the points in (2.5, 3.5). As expected, walking is true at greater proportions of intervals near the middle.

5.3 Continuous subinterval inferences

We now consider a variation of the problem dealt with in Section 5.2. Given that walking is continuous definite integral information over the interval (0,10) and that $\int_0^{10} \text{walking(t)}dt = 7$, what is the probability that walking is true throughout the interval $(0.5, 1.5)$? Note this is different from asking what proportion of points in $(0.5, 1.5)$ is walking true.

If R and L are defined as in Section 5.1, then the probability β is true throughout $(t_0, t_1) \subseteq R$ is:

$$\begin{aligned}
&p\left(\int_{t_0}^{t_1} \beta(t)dt = t_1 - t_0\right) \\
&\quad = \max\left(0, \frac{L - (t_1 - t_0)}{|R| - (t_1 - t_0)}\right).
\end{aligned}$$

This is a deductive consequence and requires no inductive assumption about probabilities at points.

Returning to the walking example, the probability that walking is true throughout $(0, 1)$ is zero, throughout $(5, 6)$ is 1.0, and throughout $(0.5, 1.5)$ is 0.05.

6 Comparison with other work

The only previous attempt at representing a subset of definite integral information is Allen's processes [Allen, 1984]. Examples of processes are 'I am walking' and 'ran a while' over some interval. Questions have been raised about the correctness of Allen's representation (see [Trudel, 1992] for a discussion). Also, Allen does not consider the types of reasoning we are interested in.

Eberbach et. al. [Eberbach and Trudel, 1992] consider a problem which can be viewed as the opposite of that dealt with here. Instead of deriving probabilities at points based on interval based information, they begin with the probabilities at the points and then derive probabilities for the interval based information.

7 Conclusion

We have shown how to represent and reason with definite integral information—approximate knowledge about the duration of temporal information. Although very little can be said that is categorically true about any subinterval or any specific point, many interesting results can be derived from the axioms of probability given the notion of inference from a maximally specific reference class. For example, given a set of temporal statistics, we have shown how to compute the probability that temporal information is true at an interior point, the probability the information persists throughout a subinterval and the expected proportion of points at which temporal information is true in a subinterval.

Acknowledgements

The first author acknowledges the support of the Institute for Robotics and Intelligent Systems (IRIS). Research of the second author was supported by IRIS and NSERC grant OGP0099045. Research of the third author was supported by NSERC grant OGP0046773. We thank Howard Hamilton for his comments.

Appendix A

The computation of the expected value of *raining* over December from the given definite integral information (see Figure 2) involves dividing December into two subintervals R_{3a} and R_{3b}. We then use Direct Inference Rule 2 in conjunction with the maximally specific reference intervals $R_5 = R_1 - R_2 - R_4$ and R_2 as follows:

By Proposition 5,

$$E\left(\frac{\int_{R_3} raining(t)dt}{|R_3|}\right) = \frac{E\left(\int_{R_3} raining(t)dt\right)}{|R_3|}.$$

Considering just the numerator,

$$E\left(\int_{R_3} raining(t)dt\right) =$$

$$E\left(\int_{R_{3a}} raining(t)dt\right) + E\left(\int_{R_{3b}} raining(t)dt\right).$$

By rewriting Proposition 7 as

$$E\left(\int_S \beta(t)dt\right) = \frac{E\left(\int_R \beta(t)dt\right)}{|R|} \cdot |S|,$$

we can then write

$$E\left(\int_{R_{3a}} raining(t)dt\right)$$

$$= \frac{E\left(\int_{R_5} raining(t)dt\right)}{|R_5|} \cdot |R_{3a}|,$$

and

$$E\left(\int_{R_{3b}} raining(t)dt\right)$$

$$= \frac{E\left(\int_{R_2} raining(t)dt\right)}{|R_2|} \cdot |R_{3b}|.$$

From previous computations, we substitute to obtain

$$E\left(\int_{R_{3a}} raining(t)dt\right) \approx 0.33 \cdot 20$$

and

$$E\left(\int_{R_{3b}} raining(t)dt\right) = 0.75 \cdot 11.$$

Then

$$E\left(\int_{R_3} raining(t)dt\right) \approx 0.33 \cdot 20 + 0.75 \cdot 11$$

and

$$E\left(\frac{\int_{R_3} raining(t)dt}{|R_3|}\right) \approx 48\%.$$

References

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

[Eberbach and Trudel, 1992]
E. Eberbach and A. Trudel. Representing spatial and temporal uncertainty. In *Fourth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Palma de Mallorca, Spain, 1992. [to appear].

[Goodwin, 1991] S.D. Goodwin. *Statistically Motivated Defaults*. PhD thesis, Department of Computing Science, University of Alberta, Edmonton, Alberta, 1991. [also University of Regina Computer Science TR-91-03].

[Kyburg, 1983] H.E. Kyburg, Jr. The reference class. *Philosophy of Science*, 50(3):374–397, September 1983.

[Pollock, 1984] J. Pollock. Foundations for direct inference. *Theory and Decision*, 17:221–256, 1984.

[Reichenbach, 1949] H. Reichenbach. *Theory of Probability*. University of California Press, Los Angeles, California, 1949.

[Trudel, 1990] A. Trudel. Temporal integration. In *Proceedings of the Eighth Biennial Conference on the Canadian Society for Computational Studies of Intelligence*, pages 40–45, Ottawa, Canada, May 1990.

[Trudel, 1992] A. Trudel. A formal specification of Allen's processes. In *European Conference on Artificial Intelligence*, Vienna, Austria, August 1992. [submitted].

A Characterization of Extensions of Gerneral Default Theory

Zhang Mingyi
Guizhou Academy of Sciences
40 East Yanan Road, Guiyang City
Guizhou Province, 550001, P.R. China

Abstract

A formalism for default reasoning introduced by Reiter [Reiter 1978a, 1978b, 1980, 1981, 1982], which provides a formal framework for an important part of human reasoning, appears to be the most stable one of the approaches to non-monotonic inference[Ginsbery, 1987]. However, Reiter and his followers have been placing a heavy emphasis on a class of so-called normal defaults since then, the reason why they do so seems to be that a general class of defaults is mathematically intractable [Reiter and Cirscuolo, 1981]. So up to now, the main point about general default theories remain unclear. In this paper a characterization of extensions of general default theories is given. Basded on this, several results on the existence of extensions are shown .All these essentially develop Reiter's theory and provide for introducing another class of defaults, so-called Auto-compatible Default Theory, which is more general than Normal Default Theory and enjoys very nice feature of the latter [Mingyi, 1991a, 1991b].

Key Words : Default Theory, Extension of Default Theories, Compatible Subset of Defaults, Auto-compatible Default.

1. Introduction

Default reasoning is foundmental in Artificial Intelligence(AI) because it appears to be the only tractable method to deal with the problem of incomplete information and it is specifically concerned with common sense reasoning, which has recently been recognized in the AI literature to be foundamental importance for knowledge representation[Besnard, 1989].

A formalism for default resoning introduced by Reiter[Reiter, 1978a], Default Logic, which puts forward the idea of using it as a knowledge represetation scheme for an important part of human reasoning in AI, appears to be the most stable: almost without exception, the articles that have appeared after Reiter's original one focus on applying Reiter's idea as opposed to modifying them. The single exception is Lukaszewicz's modification[Lukaszewicz, 1984a, 1984b, 1985]. A number of people feel unconfortable with the face that not all theories of default logic have extension and that there is no proof theory worthy of this name of the whole logic. Lukaszewicz (1984*b*) has proposed an alternative view of the nation of extensions, one by which extensions exist for any default theory. However, knowledge representation problems involving default reasoning need not be given solutions more conforming to everybody's intuitions by Lukaszewicz's modified formalization of derivability using default than by Reiter's original one. No Logic can be expected to be perfect for all cases for which it has been devised to handle. There are always examples that can be given conterintutive or paradoxical interpretations. So default Logic cannot be attacked on the grounds that no particular approach to it deserves to be preferred over all others. In fact, Reiter's formalism allows for extending variations as well as for convervative variations (e.g.Rychlik, 1985; Saint-Dizier, 1986, 1988; Etherigton, 1987; Froidevaux and Kagser, 1988; Mercer, 1987).

However, Reiter and his followers have been placed a heavy emphasis on the class of so-called normal defaults and less concerned about the general class of defaults since general default theories are mathematically intractable[Reiter and Cirscuolo, 1981]. Though it is so, Reiter et al. noticed that normal defaults can interact with each other in ways that lead to counterintutive results and had to introduce a general class of so-called semi-normal defaults. Unfortunately, the semi-normal class lacks many nice features which are enjoyed by the normal class. Up to now , we little know intersting investigation in this topic and main point about general default theories remains unclear. In paticular, the problem on the existence of extensions of general default theories, which is complex and difficult to deal with, has been not explored.

Following a differet approach from one of [Reiter, 1980], this paper will explore the above mentioned problem on the existence of extensions. Though Reiter gave a characterization of extensions (Theorem 2.1 [Reiter, 1980]), it is faced with trouble testing an enormous collection of sets of closed wffs to determine if a default theory has an extension and such a testing is computationally intractable and inconvenient to investigation. So, we give a characterization of extensions , which depends only on the given information and default rules. According to this, the sufficient conditions of [Mingyi, 1991a] becomes a direct result of the condition given by introducing the notions of compatible subsets of defaults and auto-compatible defaults in the paper. All this provides for introducing another class of defaults, so-called Auto-compatible Defaults Theory [Mingyi, 1991b], which is more general than Normal Default Theory and enjoys very nice feature of the latter. It is worth noticing that the sufficient condition on the existence and uniqueness of extension for general default theories given in [Besnard, 1989](As far as we know, it is one of a few results on the existence of extesions expect Reiter's result) is just a corollary of our result, and Theorem 3 of [Wu, 1991] is only a particular example of the result of [Besnard, 1989]. Hence, our work is an essential development of Reiter's theory.

Following [Reiter, 1980], we use the following concepts and notations.

By a default δ we mean any configuration of the form

$$\alpha(\vec{x}) : M\beta_1(\vec{x}), ...M\beta_n(\vec{x})/\gamma(\vec{x})$$

where $\alpha(\vec{x})$, $\beta_1(\vec{x})$, $...\beta_n(\vec{x})$, $\gamma(\vec{x})$ are wffs whose free variables are occur in $\vec{x} = x_1, ..., x_m$. $\alpha(\vec{x})$ is called the prerequisite of the default, $\beta_1(\vec{x}), ..., \beta_n(\vec{x})$ is called the consistency condition of the default δ, $\gamma(\vec{x})$ is called the consequent of the default δ.

For simplicity, the form of a default δ is usually abbreviated as $\alpha : M\beta_1, ..., M\beta_n/\gamma$ provided it does not sacrifice clarity.

A default is closed iff none of α, β_1, ..., β_n, γ contains a free variable.

A default theory Δ is a pair (D, W) where D is a set of defaults and W is a set of closed wffs. If every element of D is closed, then (D, W) is called a closed default theory. As in [Reiter, 1980], the technical results on closed default theories can be generalized to the case of open defaults by Skolemizing W as well as all of defaults of D. So there is clearly no loss of generality by requiring wffs of W and $CON(D)$ to be closed. According to this, this paper only disscusses the closed default theories.

Given a closed default theory $\Delta = (D, W)$ and any $D' \subseteq D$, define

$$PRE(D') = \{\alpha | (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in D'\}$$

$$CON(D') = \{\gamma | (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in D'\}$$

$$CCS(D') = \{\beta_i | (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in D' and 1 \leq i \leq n\}$$

2. Existence of Extensions

In [Reiter, 1980] a characterization of extension by means of a sequence of sets of closed wffs(Theorem 2.1)is given by which we can determine if a set E of closed wffs is an extension of a closed default theory. Based on this, a few properties of default theories and a complete theory on the class of normal defaults are obtained.However, it is very difficult to decide whether a set of closed wff is an extension of a closed default theory by the condition since it needs to test an enormous collection of sets. So this obstructs further study on the general class of default. In this section, it starts immidiately from a default theory's self to explore the problem on the existence of extensions. Unquestionably our result depending only given information W and the sets D of defaults is more simple and natural.

Definition 2.1: Let E be a set of closed wffs and $\Delta = (D, W)$ a closed default theory. Let us define

$E_0(\Delta) = W$

$E_{i+1}(\Delta) = Th(E_i(\Delta)) \cup \{\gamma | (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in D$

$$where\ \alpha \in E_i(\Delta) and \neg\beta_1, ..., \neg\beta_n \notin E\}$$

Let

$$\Theta(E, \Delta) = \bigcup_{0 \leq i < \infty} E_i(\Delta)$$

For $i > 0$, define $GD(E_i(\Delta)) = \{(\alpha : M\beta_1, ...M\beta_n/\gamma) \in D | \alpha \in E_{i-1}(\Delta) and \neg\beta_1, ..., \neg\beta_n \notin E\}$

and $GD(E_0(\Delta)) = \emptyset$. Let

$GD(\Theta(E, \Delta), \Delta) = \bigcup_{0 \leq i < \infty} GD(E_i(\Delta))$.

Usually, the symbol Δ in the notation $E_i(\Delta)$ $(i \geq 0)$ will be omitted if it does not make confusion.

Definition 2.2: For any closed default theory $\Delta = (D, W)$, define $D_0(\Delta) = \{(\alpha : M\beta_1, ..., M\beta_n/\gamma) \in D | W \vdash \alpha\}$

and for $i > 0$, $D_{i+1}(\Delta) = \{(\alpha : M\beta_1, ..., M\beta_n/\gamma) | W \cup CON(D_i(\Delta)) \vdash \alpha\}$

$$\lambda(D, \Delta) = \bigcup_{0 \leq i < \infty} D_i(\Delta)$$

As the above, the symbol Δ in $D_i(\Delta)(i \geq 0)$ and $\lambda(D, \Delta)$ will be omitted.

Notation: $\neg\beta_1, ..., \neg\beta_n \notin E$ and $S \nvdash \neg\beta_1, ..., \neg\beta_n$ is an abbreviation for

$\neg\beta_1 \notin E$, ..., $\neg\beta_n \notin E$ and $S \nvdash \neg\beta_1$, ..., $S \nvdash \neg\beta_n$, respectively.

It is easy to show that the following theorem is equivalent to Theorem 2.1 of [Reiter, 1980].

Theorem 2.1.Let E be a set of closed wffs and $\Delta = (D, W)$ a closed default theory, then E is an extension of Δ iff $E = \Theta(E, \Delta)$.

Corollary. Given a closed default theory $\Delta = (D, W)$ and a set of closed wffs, then

$$\Theta(E, \Delta) = Th(W \cup CON(GD(\Theta(E, \Delta), \Delta))).$$

Especially, if E is an extension of Δ then

$$E = Th(W \cup CON(GD(E, \Delta)).$$

Throughout the paper , $GD(\Theta(E, \Delta), \Delta)))$ is replaced by $GD(E, \Delta)$ when E is an extension of Δ.

Definition 2.3. Let $\Delta = (D, W)$ be a closed default theory. $D' \subseteq D$ is an incompatible subset of defaults with respect to Δ iff their is $\delta = (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in D'$ such that $W \cup CON(D') \vdash \neg\beta_i$ for some i, $1 \leq i \leq n$. If D' is not incompatible then call it a compatible subset of defaults with respect to Δ. A compatible subset D^* of defaults is maximally compatible subset of defaults iff $D^* \subseteq D'$ implies $D' = D^*$, where $D' \subseteq D$ is any compatible subset of defaults.

Corollary 2.3.[Reiter, 1980] Any closed default theory has a maximally compatible subset of defaults.

Proof: It is sufficient to notice that $SD = \{D' \subseteq D | D' \textit{ is compatible}\}$ is not empty since $\phi \in SD$. The corollary is proved by Zorn's Lemma.

Definition 2.4. Let $\Delta = (D, W)$ be a closed default theory. A default

$$\delta = (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in D$$

is auto-incompatible with respect to Δ iff there exists a compatible subset D' of defaults such that

$$W \cup CON(D') \nvdash \neg\beta_1, ..., \neg\beta_n$$

and

$$W \cup CON(D' \cup \{\delta\}) \vdash \neg\beta$$

for some $\beta \in CCS(D' \cup \{\delta\})$. Here D' is said to be an associated auto-incompatible subset of defaults with respect to δ. And if δ is not auto-incompatible then it is auto-compatible.

Corollary 2.4. Any closed default theory $\Delta = (D, W)$ with compatible subset D of defaults does not contain any auto-incompatible defaults.

Proof. It is clear from Definition 2.3 and 2.4.

Lemma 2.5. Let E be an extension of a default theory $\Delta = (D, W)$ then $\lambda(GD(E, \Delta)) = GD(E, \Delta)$.

Proof. From Definitions 2.1 and 2.2 we have

$$\lambda(GD(E, \Delta)) = \bigcup_{0 \leq i < \infty} (GD(E, \Delta))_i$$

$$GD(E, \Delta) = \bigcup_{0 \leq i < \infty} GD(E_i, \Delta)$$

First, we prove by induction on i that $(GD(E, \Delta))_i \subseteq GD(E_{2i+2}, \Delta)$.

Base: If $\delta = (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in (GD(E, \Delta))_0$ then $W \vdash \alpha$ *and* $\neg\beta_1, ..., \neg\beta_n \notin E$ which implies that $\alpha \in E_1$. So $\delta \in GD(E_2, \Delta)$, that is, we proved $(GD(E, \Delta))_0 \subseteq GD(E_2, \Delta)$.

Step: Assume that $(GD(E, \Delta))_i \subseteq GD(E_{2i+2}, \Delta)$. For any $\delta = (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in (GD(E, \Delta))_{i+1}$ then $W \bigcup CON(GD(E, \Delta)_i) \vdash \alpha$ and $\neg\beta_1, ..., \neg\beta_n \notin E$. By the induction assumption, $W \bigcup CON(GD(E_{2i+2}, \Delta)) \vdash \alpha$. Therefore $\alpha \in E_{2i+3}$.Hence $\delta \in GD(E_{2i+4}, \Delta)$.

The result implies that $\lambda(GD(E, \Delta)) \subseteq GD(E, \Delta)$.

Similarly, we can prove by induction that $GD(E_i, \Delta) \subseteq GD(E, \Delta)_{i-1}$ for all $i > 0$.

Base: It is clear that $GD(E_1, \Delta) \subseteq GD(E, \Delta)_0$.

Step: Assume that $GD(E_i, \Delta) \subseteq GD(E, \Delta)_{i-1}$. For any $\delta = (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in GD(E_{i+1}, \Delta)$, we have $\alpha \in E_i$ and $\neg\beta_1, ...\neg\beta_n \notin E$ .So, $\alpha \in Th(E_{i-1}) \cup CON(GD(E_i, \Delta))$. Theorefore $W \cup CON(GD(E_i, \Delta)) \vdash \alpha$. By the induction assumption,

$$W \cup CON((GD(E, \Delta))_{i-1}) \vdash \alpha$$

which yields $\delta \in (GD(E, \Delta))_i$

The result implies GD(E, Δ) $\subseteq \lambda$ (GD(E, Δ)). And all this complete the proof of the lemma.

Theorem 2.6. Any closed default has an extension iff there exists a compatible subset D^* of defaults such that

P1. $\lambda(D^*) = D^* \subseteq D$.

P2. For any $\delta = (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in D - D^*$, either $W \cup CON(D^*) \nvdash \alpha$ or $W \cup CON(D^*) \vdash \neg\beta_i$ for some i, $1 \leq i \leq n$.

Proof. The case where W is inconsistent is not considered, as it makes the theorem trivial. In fact, Δ has an inconsistent extension iff W is inconsistent by corollary 2.2 of [Reiter, 1980]. At the same time, D^* in the theorem is just empty.

(Only if part) If Δ has a consistent extension E. First we show that $GD(E, \Delta)$ is compatible. Assume the contrary holds: there is $\delta' = (\alpha' : M\beta'_1, ..., M\beta'_m/\gamma') \in GD(E, \Delta)$ such that $W \cup CON(GD(E, \Delta)) \vdash \neg\beta'_i$ for some i, $1 \leq i \leq m$. Since $E = Th(W \cup CON(GD(E, \Delta)))$ it follows that $\neg\beta'_i \in E$ and this is a contradiction. So $GD(E, \Delta)$ satisfies P1 by Lemma 2.5.

Next , if there is $\delta = (\alpha : M\beta_1, ..., M\beta_m/\gamma) \in D - GD(E, \Delta)$ such that

$$W \cup CON(GD(E, \Delta)) \vdash \alpha$$

and $W \cup CON(GD(E, \Delta)) \nvdash \neg\beta_1, ..., \neg\beta_n$ then $\alpha \in E, \neg\beta_1, ..., \neg\beta_n \notin E$. Hence $\delta \in GD(E, \Delta)$, a contradition.That is, $GD(E, \Delta)$ satisfies P2.

(If part) Assume that D^* is the compatible subset of defaults satisfing P1 an P2. Let $E = Th(W \cup CON(D^*))$. First we show by induction on i that $\Theta(E, \Delta) \subseteq E$.

Base: Obviously, $E_0 \subseteq E$.

Step: Assume that $E_i \subseteq E$. Then $Th(E_i) \subseteq E$ and $\alpha \in E, \neg\beta_1, ..., \neg\beta_n \notin E$ for any $\delta = (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in GD(E_{i+1}, \Delta)$ since $E_{i+1} = Th(E_i) \cup CON(GD(E_{i+1}, \Delta))$. So $\delta \in D^*$ (otherwise it would contradict P2). This implies $E_{i+1} \subseteq E$.

Next we show $E \subseteq \mathcal{E}(E, \Delta)$. To prove this it is sufficient to show

$D^* \subseteq GD(\Theta(E, \Delta), \Delta)$ since $E = Th(W \cup CON(D^*))$, $W \subseteq \Theta(E, \Delta)$ and

$\Theta(E, \Delta) = Th(W \cup CON(GD(\Theta(E, \Delta), \Delta)))$. In this similitude of the proof of Lemma 2.5, we show by induction on i that $D^*_i \subseteq GD(E_{2i+2}, \Delta)$.

Base: For any $\delta = (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in D^*_0$, then $W \cup CON(D^*) \not\vdash \neg\beta_1, ..., \neg\beta_n$ in view of the compatibility of D^* and $\alpha \in E_1$ since $W \vdash \alpha$. So $\delta \in E_2$.

Step: Assume $(D^*)_i \subseteq GD(E_{2i+2}, \Delta)$. Then for any $\delta = (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in (D^*)_{i+1}$, we have $W \cup CON((D^*)_i) \vdash \alpha$. By the induction assumption,

$W \cup CON(GD(E_{i+1}, \Delta)) \vdash \alpha$. So $\alpha \in E_{2i+3}$. Hence $W \cup CON(D^*) \not\vdash \neg\beta_1, ..., \neg\beta_n$ by the compatibility of D^*. This gives $\delta \in GD(E_{2i+4}, \Delta)$ which yields $(D^*)_{i+1} \subseteq GD(E_{2i+4}, \Delta)$.

The result implies that $D^* \subseteq GD(\Theta(E, \Delta), \Delta)$ since $D^* = \lambda(D^*) = \bigcup_{0 \le i \le \infty} (D^*)_i$. Therefore $E \subseteq \Theta(E, \Delta)$. So, $E = \Theta(E, \Delta)$ and E is an extension of Δ by Theorem 2.1.

Theorem 2.7. Any closed default theory has exactly an extension iff there is a compatible subset of defaults satisfing P1 and P2 in Theorem 2.6 and for any such subsets of defaults D^*, D^{**}, $W \vdash \wedge CON(D^*) \Longleftrightarrow \wedge CON(D^{**})$ always holds, where the notation $\wedge S$ means the conjunction of elements of any set S of wff, i.e. $\wedge_{\alpha \in S} \alpha$.

Proof. It is sufficient to note that $E_1 = E_2$ iff $W \vdash \wedge CON(D^*) \Longleftrightarrow \wedge CON(D^{**})$. Where $E_1 = Th(W \cup CON(D^*))$, $E_2 = Th(W \cup CON(D^{**}))$. The remains of the proof is easy by Theorem 2.6.

Lemma 2.8. Given a closed default theory $\Delta = (D, W)$ and $D', D'' \subseteq D$. If $D' \subseteq D''$ then $\lambda(D') \subseteq \lambda(D'')$. As a result $\lambda(\lambda(D')) = \lambda(D')$ for any $D' \subseteq D$.

proof. The first part of the lemma is easily obtained by induction. For the second part, it is easy to show that $\lambda(\lambda(D')) \subseteq \lambda(D')$ by Definition 2.2. Now we prove by induction on i that $(D')_i \subseteq (\lambda(D'))_i$.

Base: It is clear that $(D')_0 \subseteq (\lambda(D'))_0$.

Step: Assume $(D')_i \subseteq (\lambda(D'))_i$. Then $\delta \in \lambda(D')$ and $W \cup CON((D')_i) \vdash \alpha$ for any $\delta = (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in (D')_{i+1}$. By the induction assumption,

$$W \cup CON((\lambda(D'))_i) \vdash \alpha$$

which implies that $\delta \in (\lambda(D'))_{i+1}$. So, $(D')_{i+1} \subseteq (\lambda(D'))_{i+1}$.

All of the above shows that $\lambda(\lambda(D')) = \lambda(D')$.

The following consequence comes immidiately from Lemma 2.8.

Corollary. Let D' be a maximally compatible subset of defaults with respect to $\Delta = (D, W)$. Then $\lambda(\lambda(D')) = \lambda(D')$ and $\lambda(D')$ is also compatible.

Lemma 2.10. If a closed default theory $\Delta = (D, W)$ has an extension E, then there is a maximally compatible subset D^* of defaults with respect to Δ such that $GD(E, \Delta) = \lambda(D^*)$.

Proof. The case where E is inconsistent is not considered, as it makes the lemma trivial. By the proof of Theorem 2.6 we know that $GD(E, \Delta)$ is compatible and $GD(E, \Delta) = \lambda(GD(E, \Delta))$. So there is a maximally compatible subset D^* of defaults such that $GD(E, \Delta) \subseteq D^*$. By Lemma 2.8, $GD(E, \Delta) \subseteq \lambda(D^*)$. Let $E^* = Th(W \cup CON(D^*))$. Clearly, $E \subseteq E^*$.Now we show by induction on i that $(D^*)_i \subseteq (GD(E, \Delta))_i$, which implies that $\lambda(D^*) \subseteq \lambda(GD(E, \Delta))) = GD(E, \Delta)$ and that $\lambda(D^*) = GD(E, \Delta)$.

Base: Clearly, $(D^*)_0 \subseteq (GD(E, \Delta))_0$.

Step: Assume $(D^*)_i \subseteq (GD(E, \Delta))_i$, then $W \cup CON((D^*)_i) \vdash \alpha$ for any $\delta = (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in (D^*)_{i+1}$. By the induction assumption, $W \cup CON((GD(E, \Delta))_i) \vdash \alpha$. Since $\neg\beta_1, ..., \neg\beta_n \notin E$ by the compatibility of D^* . It follows that $\delta \in (GD(E, \Delta))_{i+1}$. Therefore $(D^*)_{i+1} \subseteq (GD(E, \Delta))_{i+1}$.

Theorem 2.11. Any closed default theory $\Delta = (D, W)$ has an extension iff there is a maximally compatible subset D^* of defaults with respect to Δ such that for any $\delta = (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in D - D^*$, either $W \cup CON(\lambda(D^*)) \not\vdash \alpha$ or $W \cup CON(\lambda(D^*)) \vdash \neg\beta_i$ for some i, $1 \le i \le n$.

The proof simply consists of an application of Theorem 2.6 and Lemma 2.10.

Remark. Theorem 2.11 provides for more simple and helpful schema to construct extensions of a closed default theory than Theorem 2.1 of [Reiter, 1980] does because it does not depend on any set E of closed wffs like one in Theorem 2.1 of [Reiter, 1980]. Assume that we can decide whether $W \cup CON(D') \vdash \alpha, \neg\beta$ hold for a given closed default theory $\Delta = (D, W)$ and any $\alpha, \beta \in PRE(D) \cup CCS(D)$. To determine if Δ has an extension we need only to test every maximally compatible subset of defaults with respect to Δ. In other words, for a given maximally compatible subset D^* if there exists $\delta = (\alpha : M\beta_1, ..., M\beta_n/\gamma) \in D - D^*$ such that $W \cup CON(\lambda(D^*)) \vdash \alpha$ and $W \cup CON(\lambda(D^*)) \not\vdash \neg\beta_1, ..., \neg\beta_n$, then reject the D^* and continue to test another maximally compatible subset of default. This testing will procced till one of the following cases occurs:

Case 1. If all maximally compatible subsets of defaults are rejected, then Δ has not any extension.

Case 2. If some maximally compatible subset D^* of defaults is not rejected, then $E = Th(W \cup CON(\lambda(D^*)))$ is one extension of Δ.

Clearly, This schema is more direct and helpful to finding extensions than one in [Reiter, 1980].

3. Several Important Consequences

By using the nercessary and sufficient condition for the existence of extensions we obtain the following sufficient conditions which seems more important than those we know up to now .

Theorem 3.1. If a closed default theory $\Delta = (D,W)$ does not contain any auto-incompatible default then it has an extension.

Proof: In the case where W is inconsistent, it is clear that D does not contain any auto-incompatible default and Δ has an inconsistent extension. So the theorem is true.

Now assume that W is consistent.First, if $\emptyset$ is a maximally compatible subset of defaults with respect to Δ then $\emptyset$ satisfies P1 and P2 in Theorem 2.6 since Δ does not contain any auto-incompatible default. Therefore Δ has an extension $E = Th(W)$ by Theorem 2.6. Next, suppose that Δ has a non-empty maximally compatible subset D^* of defaults. For every D^*, there are two cases to be considered.

Case 1. $\lambda(D^*) = D^*$. For any $\delta = (\alpha : M\beta_1, ...M\beta_n/\gamma) \in D - D^*$, $W \cup CON(D^* \cup \delta) \vdash \neg\beta$ for some $\beta \in CCS(D^* \cup \{\delta\})$ because of the maximally compatibility of D^*. If $W \cup CON(D^*) \vdash \alpha$ and $W \cup CON(D^*) \nvdash \neg\beta_1, ..., \neg\beta_n$ then δ is auto-incompatible, which results in a contradiction.Hence the conditions of Theorem 2.11 are satified and Δ has an extension.

Case 2. $\lambda(D^*) \subset D^*$. If there is another maximally compatible subset D^{**} of defaults such that $\lambda(D^*) \subseteq D^{**}$ then

1). Δ has an extension as same as in the case 1, when $\lambda(D^{**}) = D^{**}$.

2). In the case where $\lambda(D^{**}) \neq D^{**}$, consider third maximally compatible D^{***} such that $\lambda(D^{**}) \subseteq D^{***}$. In this way we obtain a sequence of $D^*, D^{**}, ...$ such that $\lambda(D^*) \subseteq \lambda(D^{**}) \subseteq \lambda(D^{***}) \subseteq ...$ let $\overline{D} = \bigcup_{0<i} \lambda(D^{i*})$, where the notation $i*$ represents i $*$'s and we have $\lambda(D^{i*}) \subset D^{i*}$. Of course it is possible that $\overline{D} = \lambda(D^{i*})$, that is , there is not any maximally compatible subset of defaults containing $\lambda(D^{i*})$ except D^*, but this does not change the following discussion.

It is easy to show that $\lambda(\overline{D}) = \overline{D}$.In fact, if $\delta = (\alpha : M\beta_1, ...M\beta_n/\gamma) \in \overline{D}$ then there is some $i > 0$ such that $\delta \in (D^{i*})$. Since $\lambda(D^{i*}) \subseteq \overline{D}$ it follows that $\lambda(D^{i*}) \subseteq \lambda(\overline{D})$ by lemma 2.8. So $\delta \in \lambda(\overline{D})$, which implies $\overline{D} \subseteq \lambda(\overline{D})$. On the other hand, it is clear that $\lambda(\overline{D}) \subseteq \overline{D}$. Hence $\lambda(\overline{D}) = (\overline{D})$.

Next, we show that $\overline{D}$ is comtatible. Assume it is not true, that is, there is $\delta = (\alpha : M\beta_1, ...M\beta_n/\gamma) \in \overline{D}$ such that $W \cup CON(\overline{D}) \vdash \neg\beta_i$ for some i, $1 \leq i \leq n$. Then there exists $j \geq 0$ such that $\delta \in \lambda(D^{j*})$ and $W \cup CON(\lambda(D^{j*})) \vdash \neg\beta_i$ by Theorem 8.9(ii) of [Monk, 1976]. This contradicts the compatibility of $\lambda(D^{j*})$.

Finally, given any $\delta = (\alpha : M\beta_1, ...M\beta_n/\gamma) \in D - \overline{D}$, if $W \cup CON(\overline{D}) \vdash \alpha$ and $W \cup CON(\overline{D}) \nvdash \neg\beta_1, ..., \neg\beta_n$, then $W \cup CON(\overline{D} \cup \{\delta\}) \vdash \neg\beta$ for some $\beta \in CCS(\overline{D} \cup \delta)$ (if not, $\overline{D} \cup \delta$ would be compatible and $\delta \in \lambda(\overline{D})$ would hold since $W \cup CON(\overline{D}) \vdash \alpha$, this gives a contradication).The above statement shows that δ is auto-incompatible, which contradicts the hypothesis of the theorem.

All this shows that $\overline{D}$ satisfies the conditions of Theorem 2.6. So Δ has an extension.

Notice that the following result is used in the above proof.

Lemma 3.2. Let $\Delta = (D,W)$ be a closed default theory and $D' \subseteq D$ any subset of defaults. Then $\delta = (\alpha : M\beta_1, ...M\beta_n/\gamma) \in \lambda(D')$ iff $W \cup CON(\lambda(D')) \vdash \alpha$.

Proof. The only if part is trivial and for the if part it is only needed to point out that there is some $i \geq 0$ such that $W \cup CON((D')_i) \vdash \alpha$ by Theorem 8.9(ii) of [Monk, 1976] and that $\delta \in (D')_{i+1}$ follows.

Theorem 3.3. Let $\Delta = (D,W)$ be a closed default theory with the compatible subset D of defaults. Then Δ has a unique extension. In particullary, if $\lambda(D) = D$, then Δ has exactly one extension $E = TH(W \cup CON(D))$.

The proof follows immediately from Corolary 2.4 and Theorem 3.1.

Corollary 3.4 [Besnard, 1989]. A closed default theory $\Delta = (D,W)$ has exactly one extension if $\{\beta_1 \wedge ... \wedge \beta_n \wedge \gamma | (\alpha : M\beta_1, ...M\beta_n/\gamma) \in D\}$ is consistent with respect to Δ.

Proof. It is sufficient to notice that the consistency of $W \cup \{\beta_1 \wedge ... \wedge \beta_n \wedge \gamma | (\alpha : M\beta_1, ...M\beta_n/\gamma) \in D\}$ implies the compatibility of D. So, the corollary is true by Theorem 3.3.

Generally, there are examples satisfing the conditions of Theorem 3.3 but not satisfing the condition of Corollary 3.4, e.g. $\Delta = (\{\frac{:A}{B}, \frac{:\neg A}{C}\}, \emptyset)$ has a unique extension $E = Th(B \wedge C)$ since $\{\frac{:A}{B}, \frac{:\neg A}{C}\}$ is compatible, but $W \cup \{A \wedge B, \neg A \wedge C\}$ is inconsistent. It is worth mentioning that Theorem 3 of [Wu, 1991] is simply a special case of Proposition 6.2.23 of [Besnard, 1989] (it is also Corollary 3.4 here). And other theorems of [Wu, 1991] are some only trivial facts. So our results are more general than ones of [Besnard, 1989] and [Wu, 1991].

Based on Theorem 3.1, we will develop a class of defaults , so-called auto-compatible theories, which is more general than one of normal defaults and which maintains very nice features enjoyed by the latter, such as semi-monotonicity and reasonably clean proof theory. Our another paper [Mingyi, 1991b] is devoted to the class of auto-compatible defaults.

References

[Reiter, 1978a] Reiter R. On Reasoning by Default, *Proc. Theoretical Issues in Natural Language Processing*, Urbana, 2, 1978a.

[Reiter, 1978b] Reiter R. *On Closed World Databases, in Logic and Databases*, Callaire H and Minker J (Eds), Plenum Press, New York, 1978b.

[Reiter, 1980] Reiter R. A Logic for Default Reasoning, *Artificial Intelligence*, Vol.13, No.1-2, 1980.

[Reiter and Cirscuolo, 1981] Reiter R.and Cirscuolo G. On Interacting Defaults, *Proc. IJCAI-81*, Vancouver. 1981.

[Reiter, 1982] Reiter R. Circumscription Implies Predicade Completion (Sometimes), *Proc. AAAI-82*, Pittsbargh, 1982.

[McCarthy, 1980] McCarthy J. Circumscription-A from of Non-monotonic Reasoning, *Artificial Intelligence*, Vol.13, 1980.

[McCarthy, 1986] McCarthy j. Application of Circumsription to Formalizing Commonsense Knowledge, *Artificial Intelligence*, 28, 1986.

[Lukaszewicz, 1984a] Lukaszewicz W. Nonmonotonic Logic for Default Theories, *Proc.ECAI-84 Pisa*, 1984a.

[Lukaszewicz, 1984b] Lukaszewicz W. Considerations on Default Logic, *Proc. Nonmonotonic Reasoning Workshop*, New Paltz, N.Y, 1984b also *Computational Intelligence*, 4, 1988.

[Lukaszewicz, 1985] Lukaszewicz W. Two Results on Default Logic, *Proc. IJCAI-85*, Los Angles, 1985.

[Ginsbery, 1987] Ginsbery M.L., *Readings in Nonmonotonic Reasoning*, Morgan Kaufmann Publishers Inc., 1987.

[Besnard, 1989] Besnard Ph. *An Introduction to Default Logic*, Spring-verlag New-York, Berlin, Heidlberg, 1989.

[Wu, 1991] Wu, Maokong Three Theorems about Default Reasoning, *Chinese Journal of Computers*, Vol.No.8.1991.

[Monk, 1976] Monk J. *Mathematical Logic*, Springer-verlag Inc., 1976.

[Mingyi, 1991a] Zhang, Mingyi, On the existence of extensions for General Default Theories, *Techenical Report*, 1991.

[Mingyi, 1991b] Zhang, Mingyi, Auto-compatible Default Theory, *Techenical Report*, 1991.

What is a Default Priority?

Craig Boutilier
Department of Computer Science
University of British Columbia
Vancouver, British Columbia
CANADA, V6T 1Z2
email: cebly@cs.ubc.ca

Abstract

The notion of default priority has played a central role in default reasoning research. We show that Pearl's Z-ranking of default rules need not always correspond to priorities. System Z can still be used for priorities, but perhaps not in the obvious manner. Rather than the Z-ranking of rules, we show that the Z-rankings of the *negations of the material counterparts of rules* correspond naturally to priorities. We also show that the priorities of default rules can be explained in terms of belief revision by appeal to the epistemic entrenchment of the material counterpart of a rule in a *theory of expectations* in the Theorist sense. Furthermore, Brewka's notion of preferred subtheories provides a means of improving on Pearl's 1-entailment, given this connection. These results are demonstrated within the modal logic CO*, a unifying framework for various types of default reasoning and belief revision.

1 Introduction

The notion of default priority has played a central role in default reasoning research. Default rules can lead to conflicting conclusions based on certain evidence, but some rules seem to naturally take priority over others. Thus, conflicts are resolved by permitting the violation of lower priority rules in order to satisfy higher priority rules, those that are deemed more important or seen as somehow providing more information.

Many mechanisms have been proposed for representing priorities in systems like default logic and circumscription. The use of semi-normal defaults has been proposed for asserting priorities in default logic (Reiter and Criscuolo 1981). McCarthy's (1986) simple abnormality theories also embody this notion through the introduction of *cancellation of inheritance axioms*, while prioritized circumscription allows the explicit expression of priorities (McCarthy 1986; Lifschitz 1985).

While these systems allow users to express priorities, nothing about these systems explains what a priority *is*, or *why* certain rules should have higher priority. In default logic and circumscription, one merely asserts the priorities of rules, and nothing constrains these rankings to respect any intuitions. Neither of these systems provides an account of naturally occurring priorities.

Various explanations of priorities rely on the notion of *specificity* (Poole 1985). Suppose we have two default rules "birds fly" and "penguins don't." If something is known to be both a bird and a penguin, the conclusions sanctioned by these rules conflict. Most accounts claim that the rule "penguins don't fly" should be applied (or has a higher priority) because penguins are a specific subclass of birds and we prefer conclusions based on more specific information. In a probabilistic setting, this corresponds to making inferences based on the narrowest *reference class* (Bacchus 1990).[1]

While specificity seems to be an appropriate criterion for deciding priority, it wasn't until conditional theories of default reasoning were developed that specificity was put on a firm semantic basis. In particular, Pearl's (1990) System Z is a natural and compelling method of assigning "priorities" to default rules, possible worlds, and arbitrary formulae. It has commonly been understood that the Z-ranking of rules corresponds to the priorities of those rules. In this paper, we will show that this is not always the case. Rather, the priorities of rules are the Z-rankings of certain formulae, the negations of the *material counterparts* of these rules.

We can explain default priorities in terms of belief revision as well. When revising a theory or set of beliefs to include some new fact (say, some new information that has been learned), we are often forced to give up some of these original beliefs if the new information is inconsistent with the theory. The *epistemic entrenchment* of beliefs in a theory determines which of these should be given up and which should be kept when conflict arises (Gärdenfors 1988). We prefer to hold on to more entrenched beliefs. Default reasoning can be viewed as the revision of a *theory of expectations* to accommodate the known facts (Gärdenfors and Makinson 1991; Boutilier 1992d). Adopting this perspective, we show that a default priority, as defined above, is nothing more than the *degree of entrenchment of the material counterpart of a rule in the theory of expectations*. We use the correspondence between defaults and expectations to

[1] However, other considerations may be involved in choosing the *appropriate* reference class (Kyburg 1983).

propose an extension of Pearl's notion of 1-entailment, based on Brewka's (1989) *preferred subtheories*, that corrects certain deficiencies in its behavior with inheritance and independent conditionals. This approximates the idea of counting "weighted rule violations" prescribed by the maximum entropy principle (Goldszmidt, Morris and Pearl 1990), and is very similar to *conditional entailment* (Geffner and Pearl 1992).

In this paper, we will develop these connections within the unifying framework of the bimodal logic CO*. In (Boutilier 1991) this logic was shown to have the power to express *normative conditionals*, statements much like default rules, and capture solutions to the problem of irrelevance. In (Boutilier 1992c) we show that CO* is the first "classical" logic of *AGM belief revision* (Gärdenfors 1988). The results of this paper indicate that default reasoning can be viewed as a form of belief revision, a connection developed much further in (Boutilier 1992d). In Section 2 we very briefly review the logic CO* and define the normative conditional $\Rightarrow$, reading $A \Rightarrow B$ as "A normally implies B." In Section 3, we discuss belief revision and epistemic entrenchment, recall some results showing how CO* can represent these concepts, and show how it is related to default reasoning. In Section 4, we discuss System Z and some results showing that Pearl's system of ε-semantics is equivalent to a fragment of CO*. In Section 5, we demonstrate that the Z-ranking of a default rule is not always equivalent to its priority, but that the Z-ranking of a certain formula is. This turns out to be precisely the degree of entrenchment of this formula in a theory of expectations. This allows us to relate 1-entailment, a form of inference based on Z-ranking, to revision. In Section 6 we again use Z-ranking to determine priorities, but use these priorities in Brewka's (1989) model of Theorist to determine a more reasonable notion of consequence extending 1-entailment.

2 The Logic CO*

In (Boutilier 1990; Boutilier 1991), we presented modal logics in which we defined a conditional connective $\Rightarrow$, reading $A \Rightarrow B$ as "A normally implies B." While the original logics are standard modal systems, the logics CO and CO* of (Boutilier 1991) are bimodal logics with considerable expressive power. They can be used to axiomatize solutions to the problem of irrelevance that have typically required extra-logical machinery. These logics can also be used to characterize the classic AGM model of revision (Boutilier 1992c) described in the next section. We recall several definitions here, but refer the reader to these papers for further motivation and details.

We now review the Kripkean possible worlds semantics for logics of normality. The bimodal logic CO is based on a standard propositional modal language (over variables $\mathbf{P}$) augmented with an additional modal operator $\breve{\Box}$. The sentence $\breve{\Box}\alpha$ is read "α is true at all *inaccessible* worlds" (in contrast to the usual $\Box\alpha$ that refers to truth at accessible worlds).

Definition 1 A *CO-model* is a triple $M = \langle W, R, \varphi \rangle$, where W is a set (of possible worlds), R is a transitive, connected[2] binary relation on W (the accessibility relation), and φ maps $\mathbf{P}$ into 2^W ($\varphi(A)$ is the set of worlds where A is true).

Satisfaction is defined in the usual way, with the truth of a modal formula at a world defined as:

1. $M \models_w \Box\alpha$ iff for each v such that wRv, $M \models_v \alpha$.
2. $M \models_w \breve{\Box}\alpha$ iff for each v s.t. not wRv, $M \models_v \alpha$.

We define several new connectives as follows: $\Diamond\alpha \equiv_{\text{df}} \neg\Box\neg\alpha$; $\breve{\Diamond}\alpha \equiv_{\text{df}} \neg\breve{\Box}\neg\alpha$; $\ddot{\Box}\alpha \equiv_{\text{df}} \Box\alpha \wedge \breve{\Box}\alpha$; and $\ddot{\Diamond}\alpha \equiv_{\text{df}} \Diamond\alpha \vee \breve{\Diamond}\alpha$. It is easy to verify that these connectives have the following truth conditions: $\Diamond\alpha$ ($\breve{\Diamond}\alpha$) is true at a world if α holds at some accessible (inaccessible) world; $\ddot{\Box}\alpha$ ($\ddot{\Diamond}\alpha$) holds iff α holds at all (some) worlds. The logic CO is based on the following axioms and inference rules, and is complete for the class of CO-models:

K $\Box(A \supset B) \supset (\Box A \supset \Box B)$

K' $\breve{\Box}(A \supset B) \supset (\breve{\Box}A \supset \breve{\Box}B)$

T $\Box A \supset A$

4 $\Box A \supset \Box\Box A$

S $A \supset \breve{\Box}\Diamond A$

H $\breve{\Diamond}(\Box A \wedge \breve{\Box}B) \supset \ddot{\Box}(A \vee B)$

Nes From A infer $\ddot{\Box}A$.

MP From $A \supset B$ and A infer B.

We often want to insist that all logically possible worlds be contained in our set of worlds W. This gives us the extension of CO called CO*.

Definition 2 A *CO*-model* is any $M = \langle W, R, \varphi \rangle$, such that M is a CO-model and $\{f : f \text{ maps } \mathbf{P} \text{ into } \{0,1\}\} \subseteq \{w^* : w \in W\}$.[3]

This class of models is characterized by the logic CO*, the smallest extension of CO containing the following:

LP $\breve{\Diamond}\alpha$ for all satisfiable propositional α.

In order to define a normative conditional, we impose the following interpretation on the accessibility relation R: world v is accessible to w (wRv) iff v is *at least as normal* as w. Thus, R is an ordering of situations respecting the degree to which an agent judges them to be "normal" or unexceptional. The truth conditions for $A \Rightarrow B$ can be stated as "In the most normal situations in which A holds, B holds as well."[4] This condition can be expressed as

$$A \Rightarrow B \equiv_{\text{df}} \ddot{\Box}\neg A \vee \breve{\Diamond}(A \wedge \Box(A \supset B))$$

[2] R is (totally) *connected* if wRv or vRw for any $v, w \in W$ (this implies reflexivity).

[3] For all $w \in W$, w^* is defined as the map from $\mathbf{P}$ into $\{0,1\}$ such that $w^*(A) = 1$ iff $w \in \varphi(A)$; in other words, w^* is the valuation associated with w.

[4] This is only a rough formulation, for it presupposes the *Limit Assumption*, which is not a property required by our definition. There need not be a set of *most* normal worlds satisfying A. The conditional $A \Rightarrow B$ is still meaningful in this circumstance (Boutilier 1991; Boutilier 1992c).

3 Revision and Entrenchment

In this section we give a sparse description of the main ideas behind belief revision, referring readers to Gärdenfors (1988) for a comprehensive presentation of, and motivation for, work in the area.

Most work on belief revision models belief sets as deductively closed sets of sentences. We will use K to denote arbitrary belief sets, and if $K = Cn(KB)$ for some finite set KB, we will usually refer to the revision of K as the revision of its base set KB. Revising a belief set K is required when new information is learned and must be accommodated with these beliefs. If $K \not\models \neg A$, learning A is relatively unproblematic. More troublesome is revision when $K \models \neg A$ as some beliefs in K must be given up. The problem is in determining which part of K to give up, as there are a multitude of choices. Choosing which of these alternative revisions is acceptable depends largely on context. Fortunately, there are some logical criteria for reducing this set of possibilities.

The main criterion for discarding some revisions in deference to others is that of *minimal change*. Informational economy dictates that as few beliefs as possible from K be discarded in order to facilitate belief in A (Gärdenfors 1988). While pragmatic considerations will often enter into these deliberations, the main emphasis of the work of Alchourrón, Gärdenfors and Makinson (1985) is in logically delimiting the scope of acceptable revisions. They propose a set of eight postulates maintained to hold for any reasonable notion of revision (see e.g. (Gärdenfors 1988)). A revision function $*$ maps a belief set K and a proposition A to another belief set K^*_A, the result of revising K by A.

While these postulates describe logical constraints on revision, it is often the case when revising K we are more willing to give up certain sentences than others. This is referred to as *epistemic entrenchment*, and we say A is no more entrenched than B ($A \leq_E B$) if we are at least as willing to give up A as B when revising theory K. In (Gärdenfors 1988) postulates for guiding the revision of K are presented that any reasonable notion of epistemic entrenchment should satisfy. He also shows that a revision operator satisfies the eight AGM postulates iff it respects the following postulates for entrenchment:

(E1) If $A \leq_E B$ and $B \leq_E C$ then $A \leq_E C$.

(E2) If $A \vdash B$ then $A \leq_E B$.

(E3) If $A, B \in K$ then $A \leq_E A \wedge B$ or $B \leq_E A \wedge B$.

(E4) If $K \neq Cn(\bot)$ then $A \notin K$ iff $A \leq_E B$ for all B.

(E5) If $B \leq_E A$ for all B then $\vdash A$.

The revision operator $*$ and the entrenchment ordering are related by the identity

$$B \in K^*_A \quad \text{iff} \quad A \supset \neg B <_E A \supset B$$

Gärdenfors and Makinson (1991) have also shown how belief revision can determine a nonmonotonic consequence relation. We describe a specific instance of this construction applied to a default theory. We can think of default rules as corresponding to *expectations* about the world. This view is adopted in Poole's (1988) Theorist framework, where default inference is effected by considering maximal subsets of such expectations (or "hypotheses") that are consistent with the known facts. Let T be a set of default rules or expectations; for instance,

$$T = \{\texttt{penguin} \supset \texttt{bird}, \texttt{bird} \supset \texttt{fly}, \texttt{penguin} \supset \neg\texttt{fly}\}.$$

If we took this theory at face value, we could never add **penguin** on threat of inconsistency. However, revising T by **penguin** allows us to give up some of the "default rules", which allow exceptions by their very nature. In general, we say that B is a nonmonotonic consequence of A (given default theory T) if $B \in T^*_A$. This is shown to be a meaningful notion (Makinson and Gärdenfors 1990), and bears a close relationship to Theorist. Unfortunately, no notion of priority or specificity is sanctioned by these considerations alone. We examine this idea that default reasoning is the process of revising such a theory of "expectations" in Section 5, and how priorities can be determined naturally.

CO* can also be used to represent AGM revision as discussed in (Boutilier 1992c). We omit details here, but note that we can define a subjunctive conditional in CO* that captures precisely the AGM revision postulates. The connection to revision is made via the *Ramsey test*, whereby a subjunctive $A > B$ is true with respect to a given state of belief K just when revising K by A results in a belief in B. Not surprisingly, the subjunctive and normative conditionals are identical in CO* and in (Boutilier 1992d) we pursue this connection, further establishing the correlation between default reasoning and belief revision that is our concern here.[5]

Naturally, we can also capture the entrenchment of beliefs in CO*. Assuming a particular theory K is determined by a CO*-model M (see (Boutilier 1992c) for details), we can define $\leq_{EM}$, the *entrenchment ordering determined by* M, as

$$B \leq_{EM} A \text{ iff } M \models \vec{\Box}(\neg A \supset \Diamond \neg B).$$

In (Boutilier 1992b) we show this ordering respects the postulates (E1) through (E5), and that any entrenchment ordering is representable in a CO*-model.

4 System Z

One problem that has plagued default reasoning is that of priorities, as manifested in the above example. Revising T by P (using P, B, and F as the obvious abbreviations) provides no guidance as to which of the two possible resulting theories should be preferred. Intuitively, we ought to give up $B \supset F$, since P is logically stronger, or *more specific* than B. That is, default $B \supset F$ has *lower priority*, or is more readily violated, than the others. Priorities have a long tradition in default reasoning (e.g., (McCarthy 1986)), but the most natural account seems to be that of Pearl.

Pearl (1990) describes a natural ordering on default rules named the *Z-ordering*, and uses this to define a nonmonotonic entailment relation, 1-entailment, put forth as an extension of ε-semantics (Pearl 1988). The default

[5] For the interested reader, we also describe how the Gärdenfors *triviality result* is avoided in (Boutilier 1992d).

rules r of (Pearl 1990) have the form $\alpha \rightarrow \beta$, where α and β are propositional. We say a valuation (possible world) w *verifies* the rule $\alpha \rightarrow \beta$ iff $w \models \alpha \wedge \beta$, *falsifies* the rule iff $w \models \alpha \wedge \neg\beta$, and *satisfies* the rule iff $w \models \alpha \supset \beta$. For any rule $r = \alpha \rightarrow \beta$, we define r^* to be its *material counterpart* $\alpha \supset \beta$. We assume that T is a finite set of such rules (and when the context is clear, we take T to refer also to the set of material counterparts).

Definition 3 (Pearl 1990) T *tolerates* $\alpha \rightarrow \beta$ iff there is some world that verifies $\alpha \rightarrow \beta$, and falsifies no rule in T; that is, $\{\alpha \wedge \beta\} \cup \{\gamma \supset \delta : \gamma \rightarrow \delta \in T\}$ is satisfiable.

Toleration can be used to define a natural ordering on default rules by partitioning T as follows:

Definition 4 (Pearl 1990) For all $i \geq 0$ we define $T_i = \{r : r \text{ is tolerated by } T - T_0 - T_1 - \cdots T_{i-1}\}$

Assuming T is ε-consistent (see below), this results in an ordered partition $T = T_0 \cup T_1 \cup \cdots T_n$. Now to each rule $r \in T$ we assign a rank (the *Z-ranking*), $Z(r) = i$ whenever $r \in T_i$. Roughly, but not precisely (see below), the idea is that lower ranked rules are more general, or have lower priority. Given this ranking, we can rank worlds according to the highest ranked rule they falsify:

$$Z(w) = \min\{n : w \text{ satisfies } r, \text{ for all } r \in T \text{ such that } Z(r) \geq n\}.$$

Again, lower ranked worlds are to be considered more normal. Now any propositional α can be ranked according to the lowest ranked world that satisfies it; that is

$$Z(\alpha) = \min\{Z(w) : w \models \alpha\}.$$

Given that lower ranked worlds are considered more normal, we can say that a default rule $\alpha \rightarrow \beta$ should hold iff the rank of $\alpha \wedge \beta$ is lower than that of $\alpha \wedge \neg\beta$. This leads to the following definition:

Definition 5 (Pearl 1990) Formula β is *1-entailed* by α with respect to T (written $\alpha \vdash_1 \beta$) iff $Z(\alpha \wedge \beta) < Z(\alpha \wedge \neg\beta)$ (where Z is determined by T).

We will see some examples of the types of conclusions sanctioned by 1-entailment in Section 6, but refer to (Pearl 1990) for further details.

The default rules ordered by Z-ranking are usually assumed to be statements of high probability, based on Pearl's (1988) ε-semantics. A set of rules T is *ε-consistent* if each rule can be consistently assigned a conditional probability no less than $1 - \varepsilon$ for arbitrary ε approaching 0. The theory T *ε-entails* a rule $\alpha \rightarrow \beta$ if the conditional probability of β given α can be made arbitrarily high merely by making the probabilities of each rule in T achieve some threshold. We can show that this logic of default rules, based on arbitrarily high probabilities, corresponds exactly to the fragment of CO* consisting of simple conditional sentences of the form $\alpha \Rightarrow \beta$. Assume T is a finite set of simple default rules.[6]

[6] We take a rule to be either $\alpha \rightarrow \beta$ or $\alpha \Rightarrow \beta$, depending on context. We also assume each antecedent α is satisfiable, for simplicity, but these results can be restated for the more general case (Boutilier 1992a). Proofs may also be found in (Boutilier 1992a).

Theorem 1 *T is CO*-consistent iff T is ε-consistent.*

Theorem 2 *$T \models_{CO^*} A \Rightarrow B$ iff T ε-entails $A \rightarrow B$.*

Semantically, the equivalence of normative conditional inference in CO* and ε-semantics can be seen by examining Pearl's (1990) construction used to determine the ε-consistency of a set of rules (see also Adams (1975)) and equating more probable worlds with more normal worlds in the sense of CO*.

As shown in (Boutilier 1991), the notion of 1-entailment can be axiomatized in CO*. As discussed there, for any theory T there exists a unique CO*-model Z_T corresponding precisely to the Z-ranking of worlds according to T (Z_T is defined by the identity vRw iff $Z(v) \geq Z(w)$). We recall that $Z_T \models A \Rightarrow B$ iff $A \vdash_1 B$, for any given T. Thus, any semantic or syntactic results regarding Z-ranking and 1-entailment can be applied to simple conditional theories in CO*. In what follows, we take default rules to be normative conditionals in CO*.

5 Priorities as Entrenchment

We saw that revision in its simplest form could not account for priorities on conflicting defaults. However, it seems clear that some notion of epistemic entrenchment could characterize this feature. Let T be a set of expectations. As in the Theorist framework, or the Gärdenfors-Makinson revision model of nonmonotonic logic, facts (nonmonotonically) derivable from premise A are those sentences (classically) derivable from $\{A\} \cup T'$, where $T' \subseteq T$ is some maximal subset of T consistent with A. This corresponds to having an initial belief set $K = Cn(T)$ and revising K with new facts A, keeping as much of K (or more precisely, T) as possible. As we saw earlier, there can be several choices for T', but some are preferable to others. In default reasoning, these preferences are expressed as priorities on default rules: certain rules should not be applied in deference to others in the case of conflict. In revision, preferences are represented by epistemic entrenchment: certain sentences (in this case defaults) are more likely to be given up when revising than others.

Given this parallel, the question remains: do priorities correspond to entrenchments? The most obvious proposal is to associate priorities of default rules with their Z-rankings. In most naturally occurring sets of defaults (or rather, most naturally occurring "examples") this proposal is adequate. However, the following example quickly shows that the Z-ranking of rules, $Z(r)$, cannot be viewed as entrenchment of the corresponding material conditionals r^* in a default theory.

Theorem 3 *Let T be a default theory with $r_1, r_2 \in T$. Let $r_1^* \leq_E r_2^*$ iff $Z(r_1) \leq Z(r_2)$. Then $\leq_E$ will not, in general, satisfy (E1)-(E5).*

Proof As a counterexample, consider T consisting of the following four rules:

r_1 : $(p \wedge q) \Rightarrow x$
r_2 : $c \Rightarrow (\neg p \vee \neg q \vee x)$
r_3 : $p \Rightarrow (\neg c \vee q)$
r_4 : $p \Rightarrow \neg x$

It is easy to show that all rules have rank 0, except r_1, which has rank 1. A verifying assignment

for r_3, r_4 is $\{p, \neg q, \neg x, \neg c\}$, and for r_2 is $\{\neg p, c\}$, while any assignment verifying r_1 must falsify r_4. On this definition of entrenchment, $r_2^* <_E r_1^*$. However, $r_1^* \vdash r_2^*$, violating postulate (E2). ■

While Z-ranking of rules is not a coherent notion of entrenchment for our theory of expectations, we observe that the Z-ranking of *formulae* does in fact satisfy postulates corresponding to the notion of *plausibility*, the dual of entrenchment (Grove 1988; Gärdenfors 1988), and leads us to the following definition. We assume a background set of rules T and propositional A, B.

Definition 6 Let T be a default theory. We say A is *no more entrenched* (with respect to T) than B (written $A \leq_{ET} B$) iff $Z(\neg A) \leq Z(\neg B)$.

Recall that Z_T is the unique CO*-model respecting the Z-ranking of worlds determined by T.

Theorem 4 *Let $\leq_{EM}$ be the entrenchment ordering determined by Z_T. Then $\leq_{ET}$ is identical to $\leq_{EM}$.*

Corollary 5 *The relation $\leq_{ET}$ satisfies (E1)-(E5).*

What does this say about priorities on default rules? Clearly a formula is less entrenched than another if the Z-ranking of its negation is less than that of the other. This means we are prepared to violate default rules according to the following definition of priorities:

Definition 7 Let T be a default theory, $r_1, r_2 \in T$. We say r_1 has *no priority over* r_2 ($r_1 \preceq r_2$) iff $Z(\neg r_1^*) \leq Z(\neg r_2^*)$. If $r_1 \preceq r_2$ and not $r_2 \preceq r_1$, then r_1 has *lower priority* than r_2 ($r_1 \prec r_2$).

This notion of priority is consistent with the view that we will satisfy defaults with higher priorities in the case of conflict when determining the consequence relation of 1-entailment, as substantiated by the following theorem.

Theorem 6 *Let T be a default theory and let $*$ be the revision function determined by the ordering of entrenchment $\leq_{ET}$. Then $A \vdash_1 B$ iff $B \in T_A^*$.*

Corollary 7 *$B \in T_A^*$ iff $A \supset \neg B <_{ET} A \supset B$*

Example Let T contain the following conditionals:

$$P \Rightarrow B,\ B \Rightarrow F,\ P \Rightarrow \neg F,\ B \Rightarrow W$$

where we read P, B, F, W and G as "penguin," "bird," "fly," "has-wings," and "green" respectively. Using CO* alone we can derive

$$B \wedge P \Rightarrow \neg F,\ F \Rightarrow \neg P,\ B \Rightarrow \neg P.$$

Using 1-entailment we can derive further:

$$\neg B \Rightarrow \neg P,\ \neg F \Rightarrow \neg B,\ G \wedge B \Rightarrow F,\ P \wedge \neg W \Rightarrow B.$$

In our theory of expectations we have, for instance, both $P \supset F$ and $P \supset \neg F$ (since $\neg P$ is also an expectation). Since $P \supset \neg F$ is more entrenched (under Z-ranking) than $P \supset F$, the latter is given up when revising our expectations to include P, or equivalently, asking for the default consequences of P. Hence, $P \vdash_1 \neg F$.

Unfortunately, certain intuitively desirable conclusions cannot be reached through 1-entailment, in particular $P \Rightarrow W$. We turn to such difficulties in the next section.

r	$B \Rightarrow F$	$P \Rightarrow \neg F$	$P \Rightarrow B$
$Z(r)$	0	1	1
$Z(\neg r^*)$	1	2	2

Figure 1: "Common" example.

r	r_1	r_2	r_3	r_4
$Z(r)$	1	0	0	0
$Z(\neg r^*)$	2	2	1	1

Figure 2: "Uncommon" example.

Thus, we see that 1-entailment can be modeled using a revision function that satisfies entrenchment of formulae respecting the Z-ordering. That the Z-ordering of the negations of material counterparts of rules determines natural priorities (as specified by the definition given above), rather than the Z-ranking of the rules themselves, should be obvious given the following corollary.

Corollary 8 *For any i, if $\{A\} \cup \{r^* : Z(r^*) \geq i\}$ is consistent, then $A \vdash_1 r^*$ for each r^* such that $Z(r^*) \geq i$.*

This shows that if the set of rules above a certain priority threshold is consistent with some set of premises A, each of these rules is "applied" when 1-entailment is used. Thus rules are satisfied according to the priority determined by the Z-ranking of their negated material counterparts; in other words, the priority determined by the degree of entrenchment of their counterparts in the theory of expectations

$$\{A \supset B : A \Rightarrow B \in T\}$$

To the extent that these priorities are representative of default priorities in general, we can state that default priorities correspond accurately to the epistemic entrenchment of the corresponding material conditionals within a default theory. Why the Z-ranking of rules doesn't correspond to priorities is demonstrated, using our previous examples, in Figures 1 and 2. In the first, we see that "common" examples of sets of rules often satisfy the property $Z(r) = Z(\neg r^*) - 1$. Whenever we ask for the consequences of α using 1-entailment, we must inspect the set of minimal (in Z-rank) α-worlds. Given this relationship between $Z(r)$ and $Z(\neg r^*)$ we notice that default rules are "given up" in the order of their Z-ranking. However, this connection does not always hold, as evidenced by Figure 2. Using the example from Theorem 3 we see that $Z(r_2) = Z(\neg r_2^*) - 2$; so, while rule r_1 has a higher Z-ranking than r_2, r_2 cannot have lower priority than r_1, since it cannot be given up (i.e., falsified) without giving up r_1. Whenever we apply r_1, rule r_2 automatically "follows". We have determined that while the Z-ranking of default rules is a natural ordering, it cannot, in general, be viewed as a priority ranking. Instead, it induces priorities, by associating ranks with formulae, priorities corresponding to the Z-ranking of the negation of the material counterparts of rules. This view of priorities is also in concordance with the notion of epistemic entrenchment of rules within a default theory, or theory of expectations.

6 Preferred Subtheories

The Z-ranking of rules provides a very natural and compelling method of determining default priorities. The preference for more specific default rules is put on a firm semantic basis in a conditional framework and Z-ranking reflects this. To take our standard set of three default rules ($P \Rightarrow B$, $B \Rightarrow F$, $P \Rightarrow \neg F$), given the knowledge $P \wedge B$, we could potentially choose to use either the second rule or the third, but not both. In CO* (indeed, in most conditional logics for default reasoning) $P \wedge B \Rightarrow \neg F$ is derivable from this rule set, indicating a preference for the third rule. This can be explained as follows: at the most normal P-worlds, B and $\neg F$ must both hold. Since the most normal $P \wedge B$-worlds cannot be *more* normal than these P-worlds, this set is also the set of most normal $P \wedge B$-worlds. Since $\neg F$ holds at each of these, the conditional holds. Of course, these cannot be the most normal B-worlds due to the constraint $B \Rightarrow F$. Thus, the most normal B-worlds must be strictly more normal than these P-worlds (hence $B \Rightarrow \neg P$ is derivable as well). The Z-ranking of a rule indicates the degree of normality of the most normal worlds that can *confirm* that rule. The rules with P in the antecedent necessarily have a higher ranking (are confirmed by less normal worlds) than those with B as a head.

The notion of 1-entailment is determined by the Z-ranking of rules, and is based on the intuition that worlds should be considered as normal as possible subject to the constraints imposed by the rules. The Z-ranking of rules induces a ranking of worlds (or ordering of normality) through the definition of $Z(w)$. While this seems to determine a reasonable notion of consequence, certain classes of examples are not treated appropriately using 1-entailment. This problem has been identified in (Goldszmidt, Morris and Pearl 1990; Geffner and Pearl 1992). One problem is with the default inheritance of properties from superclasses. The example in the last section illustrates this phenomenon. The conclusion W (wings) is not derivable given P (penguin) even though we should expect (default) transitivity through the class B (bird). This can be explained by observing that the most normal P-worlds must violate the rule $B \Rightarrow F$ and must be given a Z-rank of 1 rather than 0 (most normal). However, any world w_1 satisfying $P \wedge B \wedge \neg F \wedge \neg W$ is given the same rank as a world w_2 satisfying $P \wedge B \wedge \neg F \wedge W$. While w_1 violates both $B \Rightarrow F$ and $B \Rightarrow W$, the Z-ranking of w_1 is determined by the *maximum* rank of the set of rules it violates. Once the rank 1 rule $B \Rightarrow F$ is violated (as in w_2), violation of a further rank 1 rule $B \Rightarrow W$ (as in w_1) incurs no additional penalty. In terms of entrenchment in the induced default theory T, the expectations $P \supset W$ and $P \supset \neg W$ are equally entrenched. A related class of examples are those containing *independent conditionals*.

Example Consider two independent defaults $R \Rightarrow W$ (if it's raining I walk to school) and $F \Rightarrow M$ (if it's Friday I have a meeting). From the knowledge $R \wedge \neg W \wedge F$ one cannot conclude via 1-entailment that M is true, even though the violation of the first default, $R \wedge \neg W$, should not prevent application of the second. Since both rules have rank 0, 1-entailment assumes that violating both rules makes a world no less normal than violating one rule.

The counterintuitive results provided by 1-entailment in each of these examples is due to its insistence on making worlds as normal as possible. This ensures that a world violating many rules of a certain rank is no less normal than a world violating a single rule of that rank, as reflected in the definition of $Z(w)$. Forcing such worlds to be as normal as their less objectionable counterparts (those violating single rules) will not effect the satisfaction of each default rule: violating one rule already ensures that a world is considered abnormal and cannot be used to confirm rules of that rank. While 1-entailment considers only the "quality" of violated default rules in its ranking of worlds, it seems natural to consider also the number of violated rules. This has been suggested by Goldszmidt, Morris and Pearl (1990) in their *maximum entropy* proposal, and in Pearl and Geffner's (1992) *conditional entailment*.

In a priority-free setting Poole's (1988) Theorist framework can be viewed as counting rule violations. Given a set of default expectations D, the default conclusions based on a set of facts F are determined by adding to F some maximal subset S of D where $F \cup S$ is consistent. As described earlier, this can seen as the revision of D to include F, but unfortunately does not allow for priorities on default rules, or the entrenchment of expectations in D. Brewka (1989) has presented a simple generalization of Theorist in which the set of defaults D is partitioned to reflect priorities. Specifically

$$D = D_0 \cup D_1 \cup \cdots D_n$$

where each D_i is a set of propositional formulae (default expectations) such that the defaults in D_i are preferred to, or have priority over, those in D_j just when $i < j$. In particular, we take D_0 to be a consistent set of *premises* which will not be violated.[7] Just as Theorist takes maximal consistent subsets of D, Brewka proposes *preferred subtheories* of D:

$$S = D_0 \cup S_1 \cup \cdots S_n$$

is a preferred subtheory of D iff $D = D_0 \cup S_1 \cup \cdots S_k$ is a maximal consistent subset of $D = D_0 \cup D_1 \cup \cdots D_k$ for $1 \leq k \leq n$. In other words, we add to D_0 as many formulae from D_1 as possible without forcing inconsistency, then add to these defaults from D_2, and so on.

The problematic aspect of Brewka's theory is that little indication of the source of these priorities (the partitioning of D) is given (although he does provide a syntactic mechanism for determining specificity). In general, any partitioning in permitted even though some of these are effectively useless. For instance, placing $B \supset F$ in D_1 and $B \wedge P \supset \neg F$ in D_2 will ensure that the second default is never applied to derive $\neg F$. System Z provides a natural ranking of rules that would seem to determine just the

[7] Brewka "prohibits" D_0, maintaining that by allowing the most reliable set of formulae (premises) to be inconsistent Theorist is generalized. However, if consistent premises are not required, this is easily captured by postulating an empty set D_0 (or F in the case of Theorist).

"priorities" needed for Brewka's partitioning. However, Brewka's notion of preferred subtheory automatically accounts for the intuitive preference that as many default rules as possible of a certain priority level be applied. This is due to the maximality condition on the subsets S_i, and stands in sharp contrast with 1-entailment. It should be a simple exercise to combine the two notions.

Let T be a consistent set of default rules or conditionals in CO*, such that T is partitioned as $T = T_0 \cup T_1 \cup \cdots T_n$. Thus, the highest ranked rules have a rank of n. Roughly, these rules should have higher priority than the others, followed by rank $n-1$ rules, and so on. The corresponding set of expectations D (the material counterparts of T) should be partitioned similarly.

Definition 8 Let T be a consistent set of simple conditionals in CO* with a maximum Z-rank of n. The *Brewka theory* D_B of expectations corresponding to T is given by $D_B = D_1 \cup \cdots \cup D_{n+1}$ where

$$D_i = \{\alpha \supset \beta : \alpha \Rightarrow \beta \in T_{n+1-i}\}$$

So, e.g., D_1 consists of the counterparts of the rank n rules, while D_{n+1} corresponds to rank 0 rules.

We can also define a skeptical consequence relation, called *B-entailment*, using the Brewka theory determined by T simply by considering what holds in all preferred subtheories of D_B. If α is a premise, it should be added to D_B in the role of D_0.

Definition 9 Let T be a set of conditionals and $D_B = D_1 \cup \cdots \cup D_{n+1}$ the corresponding Brewka theory. We say α *B-entails* β with respect to T (written $\alpha \vdash_B \beta$) iff β is entailed by all preferred subtheories of (setting $D_0 = \{\alpha\}$)

$$D_{B+\alpha} = \{\alpha\} \cup D_1 \cup \cdots \cup D_{n+1}.$$

It should be clear that asking for the consequences of the theory D_B will not correspond to 1-entailment.

Example Consider theory T from our earlier example. We saw that $P \vdash_1 W$ is not sanctioned by T. The Brewka theory D_B contains two partitions: $D_1 = \{P \supset B, P \supset \neg F\}$ and $D_2 = \{B \supset F, B \supset W\}$. The unique preferred subtheory of $D_{B+\alpha}$ where $\alpha = P$ will contain all expectations except $B \supset F$. In particular, even though the expectations in D_1 cause the violation of $B \supset F$, the other expectation in D_2 is consistent and will be satisfied, unlike 1-entailment. Hence, $P \vdash_B W$ is sanctioned by T.

A similar analysis shows that B-entailment provides more intuitive results on our example containing independent conditionals.

The Z-ranking of rules can be used to determine an ordering on possible worlds that captures 1-entailment. The definition of $Z(w)$ induced by the ranking of rules is characterized by the unique CO*-model Z_T, which in turn satisfies a conditional $\alpha \Rightarrow \beta$ just when $\alpha \vdash_1 \beta$. However, it is not an intrinsic property of the Z-ranking of rules that causes the problems in 1-entailment we examined above. Rather it is the induced ranking of worlds and the model Z_T. Indeed, Z-ranking can be used to determine different orderings on worlds, or different CO*-models. In particular, we can define a ranking of worlds, or CO*-model, that satisfies the conditionals corresponding to B-entailment, thus capturing some notion of counting rule violations in CO*. We must first introduce some terminology. We assume a fixed consistent set of conditionals T throughout, partitioned as usual.

Definition 10 For any valuation (world) w, the set of rules of rank i falsified by w is denoted

$$V_w^i = \{r \in T_i : w \text{ falsifies } r\}$$

Definition 11 For valuations w, v, let

$$\max(v, w) = \max\{i : V_w^i \subset V_v^i \text{ or } V_v^i \subset V_w^i\}$$

If the set above is empty, we let $\max(v, w) = -1$.

Thus, $\max(v, w)$ denotes the highest rule ranking such that the set of rules of this rank violated by w and v are such that one set is strictly contained in the other. We will use this quantity to rank worlds. If a world v violates a rule ranked higher than any rule violated by v, then v will be considered more normal. However, if this highest rank is the same for each world, v can be considered more normal is it violates fewer (with respect to set inclusion) rules of that rank than w.

Definition 12 The *Brewka model* of T, denoted $Z_T^B = \langle W, R, \varphi \rangle$, is defined as follows: we let W and φ be as usual, capturing the set of valuations appropriate for the our propositional language; we define R as

1. If $\max(v, w) = -1$ then vRw and wRv
2. If $V_w^{\max(v,w)} \subset V_v^{\max(v,w)}$, vRw but not wRv

Proposition 9 *Z_T^B is a CO*-model.*

Theorem 10 *$Z_T^B \models \alpha \Rightarrow \beta$ iff $\alpha \vdash_B \beta$.*

Thus, the Z-ranking of rules can be used to determine a notion of entailment in CO* that differs from 1-entailment and captures the idea that as many defaults as possible should be applied within a given priority threshold, even if certain rules cannot be applied. Naturally, the results of Section 5 can be applied directly to B-entailment as they were to 1-entailment. In particular, we can define an entrenchment ordering and revision function that corresponds to the notion of revising our expectations to effect default prediction.

Definition 13 Let T be a default theory. The entrenchment ordering for the purposes of B-entailment $\leq_{EB}$ is the entrenchment ordering $\leq_{EM}$ determined by the CO*-model Z_T^B.

Proposition 11 *The relation $\leq_{EB}$ satisfies (E1)–(E5).*

Theorem 12 *Let $*$ be the revision function induced by $\leq_{EB}$, and let D be the expectation set determined by conditionals T. Then $A \vdash_B B$ iff $B \in D_A^*$ iff $A \supset \neg B <_{EB} A \supset B$.*

Naturally, the priorities of default rules reflect the entrenchment of the corresponding material counterparts or expectations. Just as Theorem 3 shows that the Z-ranking of rules does not capture priorities precisely within the context of 1-entailment, the counterexample there also shows this to be the case for B-entailment. So

while the "priority levels" of Brewka seem compelling, they do not provide a guarantee that rules (or more precisely, expectations) will be given up in the order specified by the partition. In particular, the ordering specified by the partition $D_1 \cup \cdots D_n$ will not, in general, be an entrenchment ordering; but the adopting the view that priorities correspond to entrenchment of expectations is justified, as it is quite easy to show that the analog of Corollary 8 holds for B-entailment (where Z-ranking of formulae is replaced by entrenchment using $\leq_{EB}$).

7 Concluding Remarks

We have shown that Z-ranking is a useful way of ranking rules, but that these ranks cannot generally be interpreted as priorities. Rather these induce entrenchment orderings on a theory of expectations, and revision of this theory corresponds to default prediction. Furthermore, Z-ranking need not be tied to 1-entailment, but can be used to induce priorities for other forms of entailment. We have presented one such notion, appealing to Brewka's preferred subtheories, and demonstrating its applicability on certain examples on which 1-entailment fails to behave appropriately. Brewka's model reflects many of the same intuitions as prioritized circumscription (McCarthy 1986) and our B-entailment bears a remarkable similarity to Geffner and Pearl's (1992) conditional entailment. In particular, both of these notions of consequence have the goal of minimizing violations of defaults, but prefer to satisfy any higher priority default at the expense of lower priority defaults. In circumscription, however, priorities must be specified independently.

Although we have not done so here, it should be easy to see how the Brewka model Z_T^B can be axiomatized in CO*, just as the model Z_T is axiomatized in (Boutilier 1991). However, the Theorist-style formulation suggests a straightforward conceptual and computational approach to B-entailment. While the notion of counting rule violations within a priority level is captured by B-entailment, this notion is somewhat different from the implicit "sum of weighted rule violations" of the maximum entropy formalism (Goldszmidt, Morris and Pearl 1990). Both maximum entropy and conditional entailment address certain difficulties with the priorities induced by System Z. An investigation of the differences with B-entailment should prove enlightening. The revision model of defaults might also be applied to these systems as well, 1-entailment and B-entailment simply being two examples of the use of priorities as entrenchment. This may illuminate important similarities and distinctions among these systems.

Acknowledgements

I'd like to thank Moisés Goldszmidt, Judea Pearl and David Poole for their helpful comments.

References

Adams, E. W. 1975. *The Logic of Conditionals.* D.Reidel, Dordrecht.

Alchourrón, C., Gärdenfors, P., and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530.

Bacchus, F. 1990. *Representing and Reasoning with Probabilistic Knowledge.* MIT Press, Cambridge.

Boutilier, C. 1990. Conditional logics of normality as modal systems. In *Proc. of AAAI-90*, pages 594–599, Boston.

Boutilier, C. 1991. Inaccessible worlds and irrelevance: Preliminary report. In *Proc. of IJCAI-91*, pages 413–418, Sydney.

Boutilier, C. 1992a. Conditional logics for default reasoning and belief revision. Technical Report KRR-TR-92-1, University of Toronto, Toronto. Ph.D. thesis.

Boutilier, C. 1992b. Epistemic entrenchment in autoepistemic logic. (submitted).

Boutilier, C. 1992c. A logic for revision and subjunctive queries. In *Proc. of AAAI-92*, San Jose. To appear.

Boutilier, C. 1992d. Subjunctives, normatives and autoepistemic defaults. (submitted).

Brewka, G. 1989. Preferred subtheories: An extended logical framework for default reasoning. In *Proc. of IJCAI-89*, pages 1043–1048, Detroit.

Gärdenfors, P. 1988. *Knowledge in Flux: Modeling the Dynamics of Epistemic States.* MIT Press, Cambridge.

Gärdenfors, P. and Makinson, D. 1991. Nonmonotonic inference based on expectations. To appear.

Geffner, H. and Pearl, J. 1992. Conditional entailment: Bridging two approaches to default reasoning. *Artificial Intelligence*, 53:209–244.

Goldszmidt, M., Morris, P., and Pearl, J. 1990. A maximum entropy approach to nonmonotonic reasoning. In *Proc. of AAAI-90*, pages 646–652, Boston.

Grove, A. 1988. Two modellings for theory change. *Journal of Philosophical Logic*, 17:157–170.

Kyburg, Jr., H. E. 1983. The reference class. *Philosophy of Science*, 50(3):374–397.

Lifschitz, V. 1985. Computing circumscription. In *Proc. of IJCAI-85*, pages 121–127, Los Angeles.

Makinson, D. and Gärdenfors, P. 1990. Relations between the logic of theory change and nonmonotonic logic. In Fuhrmann, A. and Morreau, M., editors, *The Logic of Theory Change*, pages 185–205. Springer-Verlag, Berlin.

McCarthy, J. 1986. Applications of circumscription to formalizing commonsense reasoning. *Artificial Intelligence*, 28:89–116.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo.

Pearl, J. 1990. System Z: A natural ordering of defaults with tractable applications to default reasoning. In Vardi, M., editor, *Proceedings of Theoretical Aspects of Reasoning about Knowledge*, pages 121–135. Morgan Kaufmann, San Mateo.

Poole, D. 1985. On the comparison of theories: Preferring the most specific explanation. In *Proc. of IJCAI-85*, pages 144–147, Los Angeles.

Poole, D. 1988. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47.

Reiter, R. and Criscuolo, G. 1981. On interacting defaults. In *Proc. of IJCAI-81*, pages 270–276, Vancouver.

Possible Worlds Semantics for Default Logics

Philippe Besnard
IRISA
Campus de Beaulieu
F–35042 Rennes Cédex
besnard@irisa.fr

Torsten Schaub
FG Intellektik, TH Darmstadt
Alexanderstraße 10
D–6100 Darmstadt
torsten@intellektik.informatik.th-darmstadt.de

Abstract

In this paper we introduce a uniform semantical framework of various default logics in terms of Kripke structures. The approach provides a simple but meaningful instrument for comparing existing default logics in a clear setting. Our possible worlds semantics is motivated by means of constrained default logic. It is then extended to Reiter's original default logic as well as Łukaszewicz' variant — whereas it easily deals with Brewka's cumulative default logic. Since the presentation is put into the perspective of "commitment to assumptions" we obtain also a very natural modal interpretation on the notion of commitment.

1 Motivation

Recent research on default logic [Reiter, 1980] has produced many derivatives of Reiter's original formalism [Łukaszewicz, 1988; Brewka, 1991; Delgrande and Jackson, 1991; Schaub, 1991b]. Among them, all variants of default logic dealing with the notion of "commitment to assumptions" [Poole, 1989] employed constraints, either on formulas [Brewka, 1991] or on sets of formulas [Delgrande and Jackson, 1991; Schaub, 1991b]. Extending the semantics for classical default logic[1] introduced in [Etherington, 1987], a semantics for those variants was given in [Schaub, 1991a] which was based on a two–folded semantical structure: A pair of classes of models $(\Pi, \breve{\Pi})$. A deductively closed set E (called extension) of possibly nonmonotonic conclusions with underlying constraints C could then be captured by such a pair $(\Pi, \breve{\Pi})$. The first class, Π, characterizes E and the second class, $\breve{\Pi}$, characterizes C.

Although the elements of those semantical structures were (classical) first order interpretations, splitting the semantical characterizations of the extension and its underlying constraints might appear to be artificial. On the other hand, Kripke structures provide means to establish relations between first order interpretations: A Kripke structure has a distinguished world, the "actual" world, and a set of worlds accessible from it (each world is a first order interpretation).

At first, we present a semantical characterization of constrained extensions [Delgrande and Jackson, 1991; Schaub, 1991b] in terms of Kripke structures, thereby avoiding two–folded semantical structures. Given a constrained extension (E,C) and a Kripke structure $\mathfrak{m}$, we demand the actual world to be a model of the extension, E, and demand each world accessible from the actual world to be a model of the constraints, C. That is, $\mathfrak{m} \models E \wedge \Box C$.[2]

The intuition behind our construction is very natural and easy to understand: The actual world of a Kripke structure exhibits what we believe and the accessible worlds exhibit what commitments we have allowed to adopt our beliefs. Hence, the actual world is our envisioning of how things are, whereas the surrounding worlds represent the context in which that envisioning takes place.

We thus obtain a very natural semantical characterization of the notion of commitment in the context of default logics. The idea is roughly as follows. Our beliefs consist of the conclusions given by the applying default rules and the constraints (or commitments) on our beliefs stem from the justifications provided by the same default rules. Discussing the notion of cumulativity, we can moreover analyze the concept of nonmonotonic lemmata in default logics.

Also, we show how our possible worlds semantics captures classical default logic and Łukaszewicz' variant. We can then easily compare default logics and characterize the differences between them, especially in terms of commitment to assumptions as indicated above. In particular, the semantics reveals that all of the various default logics employ constraints (induced by the consequents and justifications of applied default rules) but differ basically in the extent to which the constraints are taken into account.

The rest of the paper is organized as follows. In Section 2 we reproduce the basic definition of classical and constrained default logic. In Section 3 we introduce our possible worlds semantics for constrained default logic and show how it characterizes the notion of commitment. Furthermore, we show that the semantics is able to cap-

[1] For the sake of clarity, we will refer to Reiter's default logic as *classical* default logic.

[2] Given a set of formulas S let $\Box S$ stand for $\wedge_{\alpha \in S} \Box \alpha$.

ture cumulative default logic, too. Eventually, in Section 4 and 5 we demonstrate that our semantics applies to Reiter's and Łukaszewicz' versions of default logic as well.

2 From classical to constrained extensions

Classical default logic was defined by Reiter in [1980] as a formal account of reasoning in the absence of complete information. It is based on first order logic, whose sentences are hereafter simply referred to as formulas (instead of closed formulas). A *default theory* (D, W) consists of a set of formulas W and a set of *default rules* D. A default rule is any expression of the form $\frac{\alpha:\beta}{\gamma}$ where α, β and γ are formulas. α is called the *prerequisite*, β the *justification*, and γ the *consequent* of the default rule.

Default knowledge is incorporated into the framework by means of default rules as nonmonotonic inference rules. They sanction inferences that rely upon given as well as absent knowledge. Such inferences therefore could not be made in a classical framework. A default rule is applicable, if its prerequisite holds and its justification is consistent, ie. adding its negation does not yield a contradiction.

Informally, an *extension* of the initial set of facts W is defined as all formulas derivable from W using classical inference rules and all specified default rules:

Definition 2.1 *Let (D, W) be a default theory. For any set of formulas S let $\Gamma(S)$ be the smallest set of formulas S' such that*

1. $W \subseteq S'$,
2. $Th(S') = S'$,
3. *For any $\frac{\alpha:\beta}{\gamma} \in D$, if $\alpha \in S'$ and $\neg\beta \notin S$ then $\gamma \in S'$.*

A set of formulas E is a classical extension of (D, W) iff $\Gamma(E) = E$.

Classical default logic does not enjoy the desirable features known as "commitment" and "cumulativity". The case of cumulativity is postponed to a later part. First, we concentrate on the notion of commitment.

Example 2.1 (non–commitment) *The default theory $\left(\left\{\frac{:B}{C}, \frac{:\neg B}{D}\right\}, \emptyset\right)$ has only one classical extension, $Th(\{C, D\})$. Both default rules have been applied, although they have contradicting justifications. Informally, there has been no commitment to the assumption B nor $\neg B$ [Poole, 1989].*

Brewka [1991] restored commitment (and cumulativity) in default logic by strengthening the applicability condition for default rules and making the reasons for believing something explicit. In order to keep track of the assumptions, he introduced *assertions*, ie. formulas labelled with the set of justifications and consequents of the default rules that have been applied. In [Delgrande and Jackson, 1991; Schaub, 1991b], it is shown how to retain commitment (and cumulativity) while dropping the change of formulas to assertions. For the formal presentation, we focus on constrained default logic as introduced in [Schaub, 1991b][3]. The approach taken by constrained default logic relies basically on dealing with two sets of formulas of the form (E, C). A default rule is applicable if its prerequisite holds in E and its justification is consistent wrt C (compare with the case of classical default logic).

Since constrained default logic does not alter the language the notion of a default theory stays the same.

Definition 2.2 *Let (D, W) be a default theory. For any set of formulas T let $\Upsilon(T)$ be the pair of smallest sets of formulas (S', T') such that*

1. $W \subseteq S' \subseteq T'$,
2. $S' = Th(S')$ *and* $T' = Th(T')$,
3. *For any $\frac{\alpha:\beta}{\gamma} \in D$, if $\alpha \in S'$ and $T \cup \{\beta\} \cup \{\gamma\} \nvdash \bot$ then $\gamma \in S'$ and $\beta, \gamma \in T'$.*

A pair of sets of formulas (E, C) is a constrained extension of (D, W) iff $\Upsilon(C) = (E, C)$.

Clearly, constrained extensions commit to their assumptions as constrained default logic employs a much stronger consistency check than classical default logic.

Example 2.2 (commitment) *The default theory $\left(\left\{\frac{:B}{C}, \frac{:\neg B}{D}\right\}, \emptyset\right)$ has two constrained extensions, $(Th(\{C\}), Th(\{C, B\}))$ and $(Th(\{D\}), Th(\{D, \neg B\}))$.*

It turns out that the semantics proposed in [Schaub, 1991a] is appropriate to characterize constrained extensions. A preference relation wrt to a set of default rules was defined — similar to [Etherington, 1987]. Simply, pairs of classes of first order interpretations like $(\Pi, \breve{\Pi})$ — called *focused model structures* — were considered instead of classes of first order interpretations. The idea is that, for a default rule $\frac{\alpha:\beta}{\gamma}$ to "apply" wrt a pair $(\Pi, \breve{\Pi})$, its prerequisite α must be valid in Π whereas the conjunction $\beta \wedge \gamma$ of its justification and consequent must be satisfiable in $\breve{\Pi}$. Taking into account all default rules, a maximal focused model structure $(\Pi, \breve{\Pi})$ is constructed that corresponds to a constrained extension, whose constraints correspond to the focused models $\breve{\Pi}$.

Definition 2.3 *Let $\delta = \frac{\alpha:\beta}{\gamma}$ and Π be a class of first order interpretations. The order $\succcurlyeq_\delta$ on $2^\Pi \times 2^\Pi$ is defined as follows. For all $(\Pi_1, \breve{\Pi}_1), (\Pi_2, \breve{\Pi}_2) \in 2^\Pi \times 2^\Pi$ we have $(\Pi_1, \breve{\Pi}_1) \succcurlyeq_\delta (\Pi_2, \breve{\Pi}_2)$ iff*

1. $\forall \pi \in \Pi_2 . \pi \models \alpha$,
2. $\exists \pi \in \breve{\Pi}_2 . \pi \models \beta \wedge \gamma$,
3. $\Pi_1 = \{\pi \in \Pi_2 \mid \pi \models \gamma\}$,
4. $\breve{\Pi}_1 = \{\pi \in \breve{\Pi}_2 \mid \pi \models \beta \wedge \gamma\}$.

Given a set of default rules D, we define $\succcurlyeq_D$ to be the partial order obtained by unioning all orders $\succcurlyeq_\delta$ such that $\delta \in D$. Clearly, constrained default logic is directly induced by this semantics. A constrained extension (E, C) is determined as E is formed by all formulas that are valid in the class Π of a $\succcurlyeq_D$–maximal focused

[3] As constrained extensions are equivalent to [Delgrande and Jackson, 1991]'s extensions of J–default logic, all results carry over to their J– and PJ–default logic.

model structure $(\Pi, \breve{\Pi})$ whereas the constraints C consist of all formulas valid in the class $\breve{\Pi}$ (the so–called focused models).

3 A modal characterization of constrained default logic

The focused model structures suggest that the ordering induced by a default rule has a modal nature with the corresponding semantical approach being based on Kripke structures. Intuitively, a pair $(\Pi, \breve{\Pi})$ is to be rendered into a class $\mathfrak{M}$ of Kripke structures such that Π is captured by the actual worlds in $\mathfrak{M}$ and $\breve{\Pi}$ by the accessible worlds in $\mathfrak{M}$. Ie. consider a non–modal formula α: it is valid in Π iff α is valid in $\mathfrak{M}$ and it is valid in $\breve{\Pi}$ iff $\Box\alpha$ is valid in $\mathfrak{M}$.

Correspondingly, the counterpart to a maximal focused model structure happens to be a class $\mathfrak{M}$ of Kripke structures such that

$$(\{\alpha \text{ non–modal } \mid \mathfrak{M} \models \alpha\}, \{\alpha \text{ non–modal } \mid \mathfrak{M} \models \Box\alpha\})$$

forms a constrained extension of the default theory under consideration. As always, the first set establishes the extension whereas the second set characterizes its constraints.

We follow the definitions in [Bowen, 1979] of a Kripke structure (called K–model in the sequel) as a quadruple $\langle \omega_0, \Omega, \mathcal{R}, \mathcal{I} \rangle$, where Ω is a non–empty set (also called a set of worlds), $\omega_0 \in \Omega$ a distinguished world, $\mathcal{R}$ a binary relation on Ω (also called the accessibility relation) and $\mathcal{I}$ is a function that defines a first order interpretation $\mathcal{I}_\omega$ for each $\omega \in \Omega$. As usual, a K–model $\langle \omega_0, \Omega, \mathcal{R}, \mathcal{I} \rangle$ is such that the domain of $\mathcal{I}_\omega$ is a subset of the domain of $\mathcal{I}_{\omega'}$ whenever $(\omega, \omega') \in \mathcal{R}$.

Formulas in K–models are interpreted using a language enriched in the following way: in a K–model $\langle \omega_0, \Omega, \mathcal{R}, \mathcal{I} \rangle$, for each $\omega \in \Omega$, the first order interpretation $\mathcal{I}_\omega$ is extended so that for each $e \in D_\omega$ (the domain of $\mathcal{I}_\omega$), a constant $\overline{e}$ is introduced, letting $\mathcal{I}_\omega(\overline{e}) = e$. In every world ω, each term is mapped into an element of D_ω as follows: $\mathcal{I}_\omega(f(t_1, \ldots, t_n)) = (\mathcal{I}_\omega(f))(\mathcal{I}_\omega(t_1), \ldots, \mathcal{I}_\omega(t_n))$, $n \geq 0$.

Given a K–model $\mathfrak{m} = \langle \omega_0, \Omega, \mathcal{R}, \mathcal{I} \rangle$, the modal entailment relation $\omega \models \alpha$ (in $\mathfrak{m}$) is defined by recursion on the structure of α:

$\omega \models P(t_1, \ldots, t_n)$	iff	$(\mathcal{I}_\omega(t_1), \ldots, \mathcal{I}_\omega(t_n)) \in \mathcal{I}_\omega(P)$
$\omega \models \neg\alpha$	iff	$\omega \not\models \alpha$
$\omega \models \alpha \vee \beta$	iff	$\omega \models \alpha$ or $\omega \models \beta$
$\omega \models \forall x\ \alpha[x]$	iff	$\omega \models \alpha[\overline{e}]$ for all $e \in D_\omega$
$\omega \models \Box\alpha$	iff	$\omega' \models \alpha$ whenever $(\omega, \omega') \in \mathcal{R}$

We write $\mathfrak{m} \models \alpha$ if $\omega_0 \models \alpha$ (in $\mathfrak{m}$). This means that $\mathfrak{m}$ is a model of α. We denote classes of K–models by $\mathfrak{M}$. We extend the modal entailment relation $\models$ to classes of K–models $\mathfrak{M}$ and write $\mathfrak{M} \models \alpha$ to mean that each element in $\mathfrak{M}$ (that is, a K–model) entails α.

In order to characterize constrained extensions semantically, we now define a family of orders on classes of K–models. Analogously to [Etherington, 1987; Schaub, 1991a], given a default rule δ, its application conditions and the result of applying it are captured by an order $\succ_\delta$ as follows.

Definition 3.1 *Let $\delta = \frac{\alpha : \beta}{\gamma}$. Let $\mathfrak{M}$ and $\mathfrak{M}'$ be distinct classes of K–models. We define $\mathfrak{M} \succ_\delta \mathfrak{M}'$ iff*

$$\mathfrak{M} = \{\mathfrak{m} \in \mathfrak{M}' \mid \mathfrak{m} \models \gamma \wedge \Box(\gamma \wedge \beta)\}$$

and

1. $\mathfrak{M}' \models \alpha$
2. $\mathfrak{M}' \not\models \Box\neg(\gamma \wedge \beta)$

The partial order $\succ_D$ is defined analogously to that in Section 2. Moreover, we define the class of K–models associated with W as $\mathfrak{M}_W = \{\gamma \wedge \Box\gamma \mid \gamma \in W\}$ and refer to $\succ_D$–maximal classes of K–models above $\mathfrak{M}_W$ as the *preferred* classes of K–models wrt (D, W).

As for modal logic, observe that the K–models define the modal system K. It makes sense because the only property needed is distributivity for the modal operator $\Box$ to ensure that the constraints are deductively closed.

As a reminder, we give below the axiom schema (K) and inference rule (NEC) that must be added to a classical first order system in order to obtain K:

$$(K) \qquad \Box(\alpha \rightarrow \beta) \rightarrow (\Box\alpha \rightarrow \Box\beta)$$

$$(NEC) \qquad \frac{\alpha}{\Box\alpha}$$

The choice of condition 2 in Definition 3.1 is also worth discussing. At first glance, it seems more adequate to require $\mathfrak{M}' \not\models \neg\Box(\gamma \wedge \beta)$ since we want to add $\Box(\gamma \wedge \beta)$ and the condition $\mathfrak{M}' \not\models \Box\neg(\gamma \wedge \beta)$ does not a priori exclude $\mathfrak{M}' \models \neg\Box(\gamma \wedge \beta)$. We illustrate why this is needed by means of the next example.

Example 3.1 *Consider the default theory $\left(\left\{\frac{:A}{A}\right\}, \{\neg A\}\right)$. With $\mathfrak{M}_W \models \neg A$, we also have $\mathfrak{M}_W \models \Box\neg A$. But using the condition $\mathfrak{M}' \not\models \neg\Box A$ would not prevent the "application" of the only default rule.*

In the following examples, we show how preferred classes of K–models can characterize constrained extensions. At first, we give a detailed example that illustrates the main idea.

Example 3.2 *Consider the default theory $\left(\left\{\frac{A : B}{C}\right\}, \{A\}\right)$ that yields the constrained extension $(Th(\{A, C\}), Th(\{A, B, C\}))$.*

In order to characterize this semantically, we start with $\mathfrak{M}_W \models A \wedge \Box A$. Since $\mathfrak{M}_W \models A$ it remains to ensure that $\mathfrak{M}_W \not\models \Box\neg(C \wedge B)$ — which is obvious. Hence, we obtain a class of K–models $\mathfrak{M}$ such that $\mathfrak{M} \models A \wedge \Box A \wedge C \wedge \Box(C \wedge B)$. Thus, the actual worlds of our K–models satisfy the formulas of the extension $Th(\{A, C\})$ whereas the surrounding worlds additionally fulfill the constraints, ie. $Th(\{A, B, C\})$.

In order to have a comprehensive example throughout the text, we extend the above commitment example as follows.

Example 3.3 (commitment) *The default theory $\left(\left\{\frac{:B}{C}, \frac{:\neg B}{D}, \frac{:\neg D \wedge \neg C}{E}\right\}, \emptyset\right)$ has three constrained extensions: $(Th(\{C\}), Th(\{B, C\}))$, $(Th(\{D\}), Th(\{\neg B, D\}))$, and $(Th(\{E\}), Th(\{\neg D \wedge \neg C, E\}))$.*

$\mathfrak{M}_W$ is the class of all K–models and clearly, we have $\mathfrak{M}_W \not\models \Box\neg(C \wedge B)$, $\mathfrak{M}_W \not\models \Box\neg(D \wedge \neg B)$, and $\mathfrak{M}_W \not\models$

$\Box\neg(E \wedge \neg D \wedge \neg C)$. *Therefore, all of the default rules are potentially "applicable".*

Let us detail the case of the first constrained extension. We obtain a $\succ_{\{\frac{:B}{C}\}}$*-greater class*

$$\mathfrak{M} \models C \wedge \Box(C \wedge B).$$

In order to show that there is a $\succ_{\{\frac{:B}{C}, \frac{:\neg B}{D}\}}$*-greater class, we would have to show that* $\mathfrak{M} \not\models \Box\neg(D \wedge \neg B)$. *But since* $\Box(C \wedge B) \models \Box B$, *we have* $\mathfrak{M} \models \Box(B \vee \neg D)$ *that prevents us from "applying" the second default rule. Analogously, we do not obtain a* $\succ_{\{\frac{:B}{C}, \frac{:\neg D \wedge \neg C}{B}\}}$*-greater class.*

The above example is illustrated by means of some canonical K–models in Figure 1.

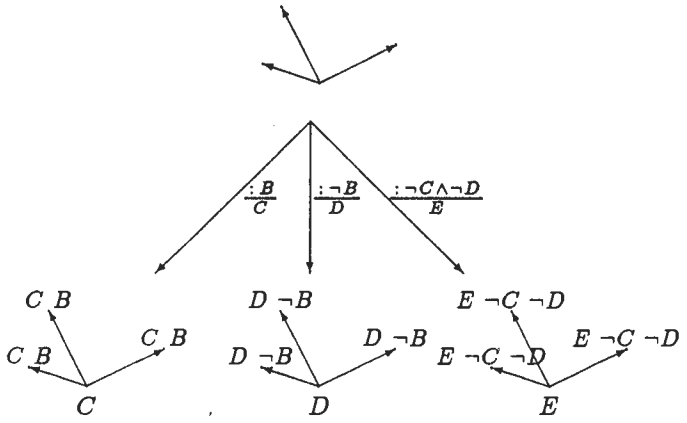


Figure 1: Commitment in constrained default logic.

The last example shows how our construction copes with self–incoherent default theories.

Example 3.4 *Consider the default theory* $(\{\frac{:\neg A}{A}\}, \emptyset)$ *whose only constrained extension is* $(Th(\emptyset), Th(\emptyset))$. $\mathfrak{M}_W$ *is the class of all K–models. But since* $\mathfrak{M}_W \models \Box\neg(A \wedge \neg A)$ *condition 2 of Definition 3.1 is falsified and, therefore,* $\mathfrak{M}_W$ *is the only preferred class.*

An interesting point concerning Definition 3.1 is that finding a non–empty $\mathfrak{M} \subseteq \mathfrak{M}'$ such that $\mathfrak{M} \models \Box(\gamma \wedge \beta)$ whenever $\mathfrak{M}' \not\models \Box\neg(\gamma \wedge \beta)$ might appear to be impossible, hence the next proposition.

Proposition 3.1 *The empty class of K–models is never preferred wrt* (D, W) *whenever* W *is consistent.*

As a corollary we obtain that the existence of constrained extensions is guaranteed.

The notion of a preferred class of K–models illustrated above is put into a precise correspondence with constrained extensions in the following theorem.

Theorem 3.2 (Correctness & Completeness) *Let* (D, W) *be a default theory. Let* $\mathfrak{M}$ *be a class of K–models and* E, C *deductively closed sets of formulas such that* $\mathfrak{M} = \{\mathrm{m} \mid \mathrm{m} \models E \wedge \Box C\}$. *Then,*

(E, C) *is a constrained extension of* (D, W) *iff* $\mathfrak{M}$ *is a* $\succ_D$*–maximal class above* $\mathfrak{M}_W$.

Then our possible worlds approach amounts to the focused model semantics [Schaub, 1991a] presented above: the first order interpretations associated with the accessible worlds take over the role of the focused models.

Corollary 3.3 *Let* (D, W) *be a default theory,* $(\Pi, \breve{\Pi})$ *a* $\succcurlyeq_D$*–maximal focused model structure above* $(\{\pi \mid \pi \models W\}, \{\pi \mid \pi \models W\})$ *and* $\mathfrak{M}$ *a preferred class of K–models wrt* (D, W). *Then, for* α, β *non–modal*

$$\Pi \models \alpha \text{ iff } \mathfrak{M} \models \alpha \text{ and } \breve{\Pi} \models \beta \text{ iff } \mathfrak{M} \models \Box\beta.$$

In the face of the above corollary, observe that a preferred class of K–models contains "more" different actual worlds than accessible ones. The reason is that focused model structures $(\Pi, \breve{\Pi})$ have the inclusion property $\breve{\Pi} \subseteq \Pi$.

How does our semantics reflect the notion of commitment? As already pointed out, the intuition behind our construction is very natural and easy to understand: The actual world of a K–model captures what we believe and the surrounding worlds capture what commitments we have allowed to adopt our beliefs. Therefore, our semantics reflects the notion of commitment through modal necessity: the commitments correspond to formulas whose necessity holds.

Since it is proved in [Schaub, 1991a] that the focused model semantics captures cumulative default logic [Brewka, 1991], Theorem 3.2 and Corollary 3.3 establish a possible worlds semantics for cumulative default logic as is shown next. First, recall that an assertion is a pair $\langle \alpha, \{\alpha_1, \ldots, \alpha_m\} \rangle$, where $\alpha, \alpha_1, \ldots, \alpha_m$ are formulas. Applied to an assertion ξ, $Form(\xi)$ gives the formula whereas $Supp(\xi)$ gives its label (called support). An assertional default theory is a pair $(D, \mathcal{W})$, where D is a set of default rules and $\mathcal{W}$ is a set of assertions.

Definition 3.2 *Let* $(D, \mathcal{W})$ *be an assertional default theory. For any set of assertions* $\mathcal{S}$ *let* $\Omega(\mathcal{S})$ *be the smallest set of assertions* $\mathcal{S}'$ *such that*[4]

1. $\mathcal{W} \subseteq \mathcal{S}'$,
2. $\widehat{Th}(\mathcal{S}') = \mathcal{S}'$,
3. *For any* $\frac{\alpha : \beta}{\gamma} \in D$, *if* $\langle \alpha, Supp(\alpha) \rangle \in \mathcal{S}'$ *and* $Form(\mathcal{S}) \cup Supp(\mathcal{S}) \cup \{\beta\} \cup \{\gamma\} \not\vdash \bot$ *then* $\langle \gamma, Supp(\alpha) \cup \{\beta\} \cup \{\gamma\} \rangle \in \mathcal{S}'$.

A set of assertions $\mathcal{E}$ *is an assertional extension for* $(D, \mathcal{W})$ *iff* $\Omega(\mathcal{E}) = \mathcal{E}$.

Now, [Schaub, 1992] shows that if (E, C) is a constrained extension of (D, W) then there is an assertional extension $\mathcal{E}$ of $(D, \{\langle \alpha, \emptyset \rangle \mid \alpha \in W\})$ such that $E = Form(\mathcal{E})$ and $C = Th(Form(\mathcal{E}) \cup Supp(\mathcal{E}))$ and conversely, if $\mathcal{E}$ is an assertional extension of $(D, \{\langle \alpha, \emptyset \rangle \mid \alpha \in W\})$ then $(Form(\mathcal{E}), Th(Form(\mathcal{E}) \cup Supp(\mathcal{E})))$ is a constrained extension of (D, W). Consequently, our possible worlds semantics also characterizes cumulative default logic:

Theorem 3.4 *Let* $(D, \mathcal{W})$ *be an assertional default theory. Let* $\mathfrak{M}_{\mathcal{W}}$ *be the class of all K–models of* $\{v \wedge \Box\eta \mid v \in Form(\mathcal{W}), \eta \in Supp(\mathcal{W})\}$. *Then, there exists a set of assertions* $\mathcal{E}$ *which is an assertional extension of* $(D, \mathcal{W})$ *such that* $\mathfrak{M} = \{\mathrm{m} \mid \mathrm{m} \models Form(\mathcal{E}) \wedge \Box Supp(\mathcal{E})\}$ *iff* $\mathfrak{M}$ *is a preferred class of K–models above* $\mathfrak{M}_{\mathcal{W}}$.

[4] $\widehat{Th}$ denotes the assertional theory operator such that if $\xi_1, \ldots, \xi_n \in \widehat{Th}(\mathcal{S})$ and $Form(\xi_1), \ldots, Form(\xi_n) \vdash \gamma$ then $\langle \gamma, Supp(\xi_1) \cup \ldots \cup Supp(\xi_n) \rangle \in \widehat{Th}(\mathcal{S})$.

In the context of cumulative default logic, naturally the question arises how the notion of cumulativity can be characterized by our possible worlds semantics. Intuitively, cumulativity demands that the addition of a theorem to the premises should not alter the set of conclusions. Apart from its theoretical interest, cumulativity is of great practical importance. Once a theory operator is cumulative, it allows for the generation of lemmata which often leads to the reduction of computational efforts.

First, let us look at the failure of cumulativity in classical default logic:

Example 3.5 (non–cumulativity[5]) *The default theory* $\left(\left\{\frac{:A}{A}, \frac{A\vee B:\neg A}{\neg A}\right\}, \emptyset\right)$ *has one classical extension,* $Th(\{A\})$. *This extension inevitably contains* $A \vee B$.

Adding this nonmonotonic theorem to the premises yields the default theory $\left(\left\{\frac{:A}{A}, \frac{A\vee B:\neg A}{\neg A}\right\}, \{A \vee B\}\right)$ *that has now two extensions:* $Th(\{A\})$ *and* $Th(\{\neg A, B\})$. *Regardless of whether or not we employ a skeptical or a credulous notion of theory formation — in both cases we are changing the set of conclusions.*

How assertional default theories restore cumulativity is shown below.

Example 3.6 (cumulativity) *The assertional default theory* $\left(\left\{\frac{:A}{A}, \frac{A\vee B:\neg A}{\neg A}\right\}, \emptyset\right)$ *has also one extension which contains the assertions* $\langle A, \{A\}\rangle$ *and* $\langle A \vee B, \{A\}\rangle$.

Adding the assertion $\langle A \vee B, \{A\}\rangle$ *to the premises yields the assertional default theory* $\left(\left\{\frac{:A}{A}, \frac{A\vee B:\neg A}{\neg A}\right\}, \{\langle A \vee B, \{A\}\rangle\}\right)$ *that has still the same assertional extension and no other.*

In the case of constrained default logic, cumulativity was preserved in [Schaub, 1991b] by means of *lemma default rules* which are prerequisite–free default rules whose justification consists of the assumptions underlying the actual lemma which is given in the consequent. The major difference between the addition of assertions to the facts and the addition of lemma default rules to the set of default rules is that once we have added an assertion to the premises it is not retractable any more whenever an inconsistency arises. Thus, the addition of assertions is stronger than that of lemma default rules. Adding an assertion to the premises eliminates all extensions inconsistent with the asserted formula or even its support. On the contrary, lemma default rules preserve all extensions and, therefore their purpose is more an abbreviation of default proofs.

How can those differences be envisioned by our semantics? Assume we have a constrained extension (E, C) and the corresponding assertional extension $\mathcal{E}$. Whenever we have a theorem $\ell \in E$ and a minimal set of default rules $D_\ell \subseteq GD_D^{(E,C)}$ $\left(= \left\{\frac{\alpha:\beta}{\omega} \;\middle|\; \alpha \in E,\ C \cup \{\beta\} \cup \{\omega\} \not\vdash \bot\right\}\right)$ which has been used to derive ℓ, there exists as well an assertion[6] $\xi_\ell = \langle \ell, \bigcup_{\delta\in D_\ell}\{Jus(\delta), Con(\delta)\}\rangle \in \mathcal{E}$. For a complement, the corresponding lemma default rule is

$$\delta_\ell = \frac{: \bigwedge_{\delta\in D_\ell} Jus(\delta) \wedge Con(\delta)}{\ell}.$$

Take a default theory (D, W) and its assertional counterpart $(D, \mathcal{W})$, where $\mathcal{W} = \{\langle \alpha, \emptyset\rangle \mid \alpha \in W\}$. Looking at cumulative default logic, we enforce (by adding the assertion ξ_ℓ to $\mathcal{W}$) that all preferred classes of K–models entail the formula

$$\ell \wedge \Box\big(\ell \wedge \bigwedge_{\delta\in D_\ell} Jus(\delta) \wedge Con(\delta)\big). \qquad (1)$$

In constrained default logic the addition of the lemma default rule δ_ℓ to the set of default rules only demands the expression (1) to be entailed by those preferred classes of K–models, to which generation the lemma default rule has contributed. That is, we enforce the entailment of (1) only for all preferred classes of K–models $\mathfrak{M}$ for which $\mathfrak{M} \succ_{GD_D^{(E,C)}\cup\{\delta_\ell\}} \mathfrak{M}_W$ holds.

4 A modal characterization of classical default logic

The possible worlds approach to default logic presented above turns out to be very general. The first evidence of this arises from the fact that the above semantical characterization carries over easily to classical default logic. Indeed, the analogue to Definition 3.1 can be defined as follows.[7]

Definition 4.1 *Let* $\delta = \frac{\alpha:\beta}{\gamma}$. *Let* $\mathfrak{M}$ *and* $\mathfrak{M}'$ *be distinct classes of K–models. We define* $\mathfrak{M} >_\delta \mathfrak{M}'$ *iff*

$$\mathfrak{M} = \{\mathfrak{m} \in \mathfrak{M}' \mid \mathfrak{m} \models \gamma \wedge \Box\gamma \wedge \Diamond\beta\}$$

and

1. $\mathfrak{M}' \models \alpha$
2. $\mathfrak{M}' \not\models \Box\neg\beta$

The partial order $>_D$ is defined analogously to that in Section 2.

Even though classical default logic does not employ explicit constraints, there is a natural counterpart given by the justifications of the generating default rules over a set of formulas E:

$$C_E = \left\{\beta \;\middle|\; \frac{\alpha:\beta}{\gamma} \in D,\ \alpha \in E, \neg\beta \notin E\right\}^8$$

We obtain a semantical characterization that yields a one–to–one correspondence between consistent extensions and non–empty $>_D$–preferred classes of K–models (an inconsistent extension trivially corresponds to $\mathfrak{M}_W$ being preferred while being empty).

Theorem 4.1 (Correctness & Completeness) *Let* (D, W) *be a default theory. Let* $\mathfrak{M}$ *be a class of K–models and* E *be a deductively closed set of formulas such that* $\mathfrak{M} = \{\mathfrak{m} \mid \mathfrak{m} \models E \wedge \Box E \wedge \Diamond C_E\}$. *Then,*

E is a consistent classical extension of (D, W) iff $\mathfrak{M}$ is a $>_D$–maximal non–empty class above $\mathfrak{M}_W$.

[5] This example is originally due to David Makinson [1989].

[6] Applied to a default rule δ, $Jus(\delta)$ yields its justification whereas $Con(\delta)$ returns its consequent.

[7] Given a set of formulas S let $\Diamond S$ stand for $\wedge_{\alpha\in S}\Diamond\alpha$.

[8] Observe that the membership qualifying property is exactly the third condition in the definition of a classical extension.

Comparing Definition 4.1 with Definition 3.1, we observe two basic differences, reflecting the fact that constrained default logic employs a stronger consistency check than classical default logic. For one thing, the second condition on $\mathfrak{M}'$ is weakened such that only β instead of $\gamma \wedge \beta$ is required to be satisfied by some accessible world of some K–model in $\mathfrak{M}'$. Notice that dropping the requirement on $\gamma \wedge \beta$ makes default logic losing the property of existence of extensions. For another thing, Definition 4.1 requires $\Diamond\beta$ to be valid in $\mathfrak{M}$ whereas Definition 3.1 requires $\Box\beta$ to be valid in $\mathfrak{M}$. Stated otherwise, the possible worlds semantics for classical extensions requires only *some* accessible world satisfying the justification β whereas the semantics for constrained default logic requires *all* accessible worlds to satisfy β.

The conclusion is that from the perspective of commitment, constrained extensions adopt their beliefs by committing to all consequents and all justifications of applied default rules whereas classical default logic commits to consequents taken together but only to justifications taken separately.

Example 4.1 (non–commitment) *The default theory* $\left(\left\{\frac{:B}{C}, \frac{:\neg B}{D}, \frac{:\neg D \wedge \neg C}{E}\right\}, \emptyset\right)$ *has only one classical extension:* $Th(\{C, D\})$.

$\mathfrak{M}_W$ *is the class of all K–models and clearly, we have* $\mathfrak{M}_W \not\models \Box\neg B$, $\mathfrak{M}_W \not\models \Box\neg(\neg B)$, *and* $\mathfrak{M}_W \not\models \Box\neg(\neg D \wedge \neg C)$. *That is, all of the default rules are potentially "applicable".*

From $\mathfrak{M}_W$ *we can construct a class of K–models* $\mathfrak{M}$ *such that* $\mathfrak{M} >_{\{\frac{:B}{C}\}} \mathfrak{M}_W$ *and*

$$\mathfrak{M} \models C \wedge \Box C \wedge \Diamond B.$$

Accordingly, we can also construct a class of K–models $\mathfrak{M}'$ *such that* $\mathfrak{M}' >_{\{\frac{:B}{C}, \frac{:\neg B}{D}\}} \mathfrak{M}_W$ *and*

$$\mathfrak{M}' \models C \wedge \Box C \wedge \Diamond B \wedge D \wedge \Box D \wedge \Diamond\neg B.$$

But it is impossible to obtain a class $\mathfrak{M}''$ *such that* $\mathfrak{M}'' >_{\{\frac{:B}{C}, \frac{:\neg B}{D}, \frac{:\neg D \wedge \neg C}{E}\}} \mathfrak{M}_W$ *since* $\mathfrak{M}' \models \Box\neg(\neg D \wedge \neg C)$.

From $\mathfrak{M}_W$, *selecting first the third default rule leads to a* $>_{\{\frac{:\neg D \wedge \neg C}{E}\}}$*–greater class*

$$\dot{\mathfrak{M}} \models E \wedge \Box E \wedge \Diamond(\neg D \wedge \neg C).$$

From $\dot{\mathfrak{M}}$ *we can construct a class of K–models* $\ddot{\mathfrak{M}}$ *such that* $\ddot{\mathfrak{M}} >_{\{\frac{:\neg D \wedge \neg C}{E}, \frac{:B}{C}\}} \mathfrak{M}_W$ *and*

$$\ddot{\mathfrak{M}} \models E \wedge \Box E \wedge \Diamond(\neg D \wedge \neg C) \wedge C \wedge \Box C \wedge \Diamond B.$$

So, $\ddot{\mathfrak{M}}$ *is the empty set of K–models because* $\Diamond(\neg D \wedge \neg C) \models \Diamond\neg C$ *and* $\Box C \wedge \Diamond\neg C \models \bot$.

The last example is illustrated by means of some canonical K–models in Figure 2.

In contrast to Proposition 3.1, the possible worlds semantics for classical default logic admits the empty set of K–models above some non–empty $\mathfrak{M}_W$. This is the case whenever a default rule is applied whose consequent contradicts the justification of some default rule which is itself applied. In particular, this reflects the failure

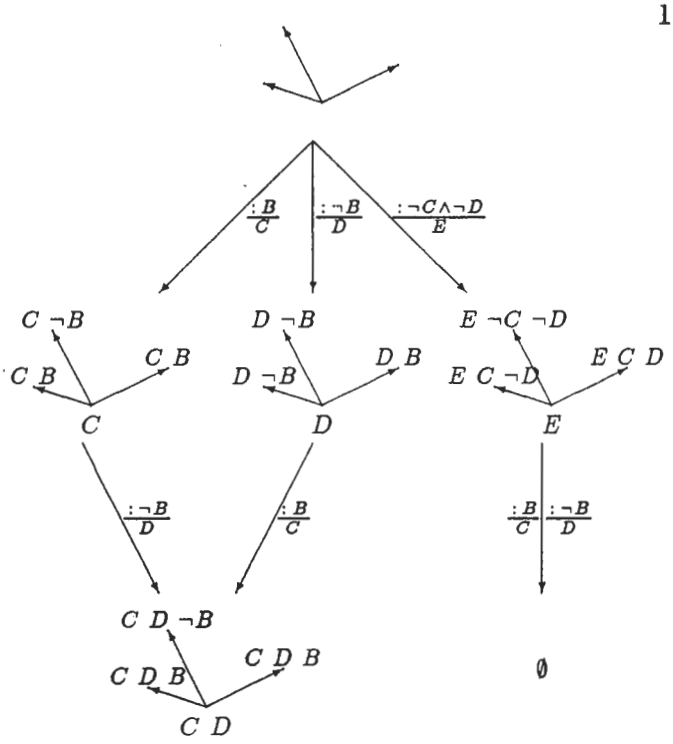


Figure 2: Commitment in classical default logic.

of semi–monotonicity in classical default logic whereas constrained default logic enjoys semi–monotonicity (A default logic is said to be semi–monotonic iff enlarging the set of default rules of a default theory can only preserve or enlarge the existing extensions.).

In addition, characterizing extensions in default logic strictly by non–empty $>_D$–maximal elements above $\mathfrak{M}_W$ avoids post–filtering mechanisms such as the *stability* criterion introduced in [Etherington, 1987]. Whenever an incoherent default theory arises, our characterization yields an empty set of K–models.

Example 4.2 *The incoherent default theory* $\left(\left\{\frac{:\neg A}{A}\right\}, \emptyset\right)$ *of Example 3.4 has no classical extension.* $\mathfrak{M}_W$ *is the class of all K–models. Clearly,* $\mathfrak{M}_W \not\models \Box A$ *but the resulting class* $\{\mathfrak{m} \in \mathfrak{M}_W \mid \mathfrak{m} \models A \wedge \Box A \wedge \Diamond\neg A\}$ *is obviously empty.*

Finally, let us examine the failure of cumulativity in classical default logic. In Section 3 we have characterized by means of a modal expression the solutions preserving cumulativity. Taking the expression given in (1) but dropping the requirement of joint consistency yields the following modal expression for classical default logic:

$$\ell \wedge \Box\left(\ell \wedge \textstyle\bigwedge_{\delta \in D_\ell} Con(\delta)\right) \wedge \Diamond Jus(D_\ell) \qquad (2)$$

where ℓ is an element of an extension E of a default theory (D, W) and $D_\ell \subseteq GD_D^{(E,E)}$ is a set of default rules used to derive ℓ.

Let us look at the canonical cumulativity example.

Example 4.3 (non–cumulativity) *Consider the default theory* $\left(\left\{\frac{:A}{A}, \frac{A \vee B : \neg A}{\neg A}\right\}, \{A \vee B\}\right)$ *obtained from Example 3.5 after adding* $A \vee B$ *(so that we are considering* $\ell = A \vee B$*). In addition to the extension*

$Th(\{A\})$, we have obtained a second one: $Th(\{\neg A, B\})$. The semantical characterization of the classical extension $Th(\{\neg A, B\})$ yields a class of K–models $\mathfrak{M}$ that is $>_{\{\frac{A\vee B\,:\,\neg A}{\neg A}\}}$–greater than $\mathfrak{M}_W$ such that

$$\mathfrak{M} \models (A \vee B) \wedge \Box(A \vee B) \wedge \neg A \wedge \Box\neg A$$

Since $D_\ell = \{\frac{:A}{A}\}$, $\mathfrak{M}$ obviously does not entail our above modal expression (2):

$$\mathfrak{M} \not\models (A \vee B) \wedge \Box((A \vee B) \wedge A) \wedge \Diamond A.$$

The entailment of the expression (2) in all preferred classes of K–models $\mathfrak{M}$ such that $\mathfrak{M} >_{GD_D^{(B,B)}\cup\{\zeta_\ell\}} \mathfrak{M}_W$ can be enforced through the corresponding lemma default rule for classical default logic (cf. [Schaub, 1992]): Given ℓ and $D_\ell = \{\delta_1, \ldots, \delta_n\} \subseteq GD_D^{(E,E)}$ as described above, we obtain

$$\zeta_\ell = \frac{: Jus(\delta_1) \wedge \bigwedge_{\delta\in D_\ell} Con(\delta), \ldots, Jus(\delta_n) \wedge \bigwedge_{\delta\in D_\ell} Con(\delta)}{\ell}.$$

5 A modal characterization of justified default logic

Further evidence for the generality of our approach is that it can easily capture a variant of default logic due to [Łukaszewicz, 1988], which we refer to as justified default logic. Indeed, the analogue to Definition 3.1 and 4.1 can be defined as follows.

Definition 5.1 *Let $\delta = \frac{\alpha:\beta}{\gamma}$. Let $\mathfrak{M}$ and $\mathfrak{M}'$ be distinct classes of K–models. We define $\mathfrak{M} \rhd_\delta \mathfrak{M}'$ iff*

$$\mathfrak{M} = \{\mathfrak{m} \in \mathfrak{M}' \mid \mathfrak{m} \models \gamma \wedge \Box\gamma \wedge \Diamond\beta\}$$

and

1. $\mathfrak{M}' \models \alpha$
2. $\mathfrak{M}' \not\models \Box\neg\beta \vee \Diamond\neg\gamma$

The partial order $\rhd_D$ is defined analogously to that in Section 2.

Compared to the order $>_\delta$ given for classical default logic, the only difference is that the condition $\mathfrak{M}' \not\models \Box\neg\beta$ has become $\mathfrak{M}' \not\models \Box\neg\beta \vee \Diamond\neg\gamma$, that is, $\mathfrak{M}' \not\models \neg(\Box\gamma \wedge \Diamond\beta)$. Indeed, the definition reveals the fact that the same constraints implicitly used in classical default logic (in the form of C_E) are explicitly attached to justified extensions[9] (in the form of J, see below) and, moreover, considered when checking consistency. That is, semantically classical and justified default logic account for the justifications of the applied default rules in form of the modal propositions $\Diamond\beta$. However, in classical default logic they are discarded when checking consistency.

Formally, a justified extension is defined as follows.

Definition 5.2 *Let (D, W) be a default theory. For any pair of sets of formulas (S, T) let $\Psi(S, T)$ be the pair of smallest sets of formulas S', T' such that*

1. $W \subseteq S'$,
2. $Th(S') = S'$,
3. *For any $\frac{\alpha:\beta}{\gamma} \in D$, if $\alpha \in S'$ and $\forall\eta \in T \cup \{\beta\}$. $S \cup \{\gamma\} \cup \{\eta\} \not\vdash \bot$ then $\gamma \in S'$ and $\beta \in T'$.*

A set of formulas E is a justified extension of (D, W) wrt to a set of formulas J iff $\Psi(E, J) = (E, J)$.

Łukaszewicz has shown in [1988] that justified default logic guarantees the existence of extensions. Semantically, requiring $\mathfrak{M}' \not\models \neg(\Box\gamma \wedge \Diamond\beta)$ and adding those K–models entailing $\Box\gamma \wedge \Diamond\beta$ makes it obviously impossible to obtain the empty set of K–models (hence the analogue to Proposition 3.1 trivially holds). Łukaszewicz has also shown that his variant enjoys semi–monotonicity. In fact, "applying" a default rule $\frac{\alpha:\beta}{\gamma}$ enforces all $\rhd_D$–greater classes of K–models $\dot{\mathfrak{M}}$ to entail $\Box\gamma \wedge \Diamond\beta$. Therefore, a later "application" of a default rule $\frac{\alpha':\beta'}{\gamma'}$ whose consequent γ' contradicts β (eg. $\gamma' = \neg\beta$) is prohibited since its "application" requires $\dot{\mathfrak{M}} \not\models \Box\neg\beta' \vee \Diamond\neg\gamma'$.

Analogously to classical default logic, Definition 5.1 only requires $\Diamond\beta$ to be valid in $\mathfrak{M}$ which is not enough for justified default logic to commit to its assumptions.

Example 5.1 (non–commitment) *The default theory $(\{\frac{:B}{C}, \frac{:\neg B}{D}, \frac{:\neg D\wedge\neg C}{E}\}, \emptyset)$ has two justified extensions, $Th(\{C, D\})$ wrt $\{B, \neg B\}$ and $Th(\{E\})$ wrt $\{\neg D \wedge \neg C\}$.*

The first one is obtained analogously to that in Example 4.1. That is, we obtain a preferred class

$$\mathfrak{M}' \models C \wedge \Box C \wedge \Diamond B \wedge D \wedge \Box D \wedge \Diamond\neg B.$$

Also, selecting first the third default rule leads to a class $\mathfrak{M} \rhd_{\{\frac{:\neg D\wedge\neg C}{E}\}} \mathfrak{M}_W$ such that

$$\dot{\mathfrak{M}} \models E \wedge \Box E \wedge \Diamond(\neg D \wedge \neg C).$$

Since we have $\dot{\mathfrak{M}} \models \Diamond\neg C$ and $\dot{\mathfrak{M}} \models \Diamond\neg D$ none of the other default rules is "applicable". Therefore, $\dot{\mathfrak{M}}$ is a (non–empty) preferred class.

Again, the last example is illustrated by means of canonical K–models in Figure 3.

Similarly to the case of classical default logic, there is a natural account of constraints attached to a set of formulas E justified by J: the justifications of the generating default rules over E, as determined by J, which are simply[10] $C_{(E,J)} =$

$$\left\{\beta \;\middle|\; \tfrac{\alpha:\beta}{\gamma} \in D,\ \alpha \in E, \forall\eta \in J \cup \{\beta\}.\ E \cup \{\gamma\} \cup \{\eta\} \not\vdash \bot\right\}$$

Then, correctness and completeness hold as in the former sections.

Theorem 5.1 (Correctness & Completeness) *Let (D, W) be a default theory. Let $\mathfrak{M}$ be a class of K–models, E a deductively closed set of formulas, and J a set of formulas such that $J = C_{(E,J)}$ and $\mathfrak{M} = \{\mathfrak{m} \mid \mathfrak{m} \models E \wedge \Box E \wedge \Diamond C_{(E,J)}\}$. Then,*

[9] Originally, Łukaszewicz called his extensions *modified* extensions.

[10] Observe that the membership qualifying property is exactly the third condition in the definition of a justified extension.

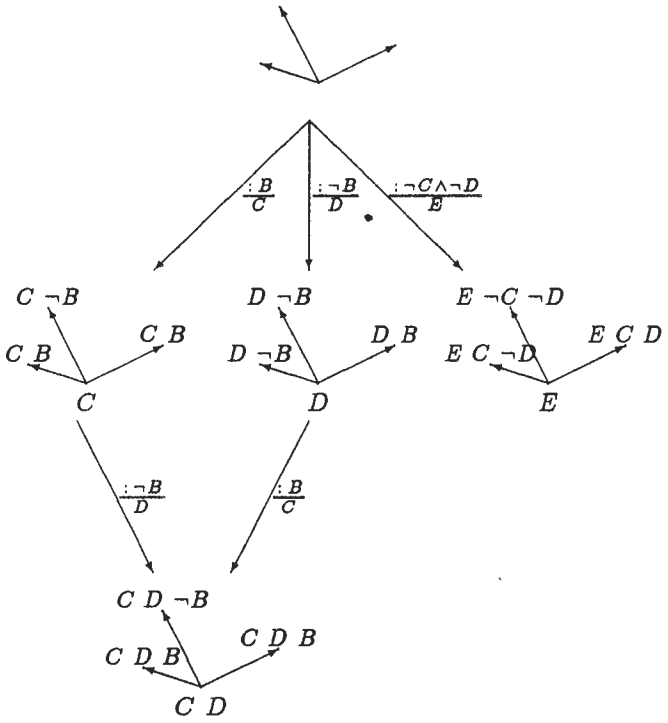


Figure 3: Commitment in justified default logic.

E is a justified extension of (D, W) wrt J iff $\mathfrak{M}$ is a $\rhd_D$-maximal class above $\mathfrak{M}_W$.

The equality $J = C_{(E,J)}$ simply states that the implicit constraints $C_{(E,J)}$ and the explicit constraints J coincide.

Notably, our possible worlds semantics is the first semantical characterization of justified default logic which is purely model-theoretic. In [1988], Łukaszewicz had to characterize justified extension by means of pairs (Π, J), where Π is a class of first order interpretations and J is a set of formulas. The reason why Łukaszewicz did so is that justified default logic allows for inconsistent sets of individually consistent constraints (so that the focussed models semantics cannot be adapted there).

6 Conclusion

We have presented a uniform semantical framework of various default logics in terms of Kripke structures. That is, we have first introduced a possible worlds semantics for constrained default logic and we have proved that it also captures cumulative default logic. Then, we have provided a simple modification for that possible worlds semantics in order to characterize classical default logic and in turn Łukaszewicz' variant.

Via adopting the perspective of commitment we have not only gained a clear criterion on that notion itself but, furthermore, provided a very natural modal interpretation by which existing default logics can be compared in a simple but deeply meaningful manner. In particular, the semantics has revealed that all of the various default logics employ constraints but differ in the extent to which the constraints are considered when checking consistency. Notably, in terms of modalities we have to switch from $\Diamond$ to $\Box$ whenever we want to preserve "commitment to assumptions".

Acknowledgements

We would like to thank Wolfgang Bibel, Gerd Brewka and Yves Moinard for useful comments on earlier drafts of this paper. In particular we are indebted to Robert Mercer for all his help. This research was partially supported by CNRS under PRC IA and by the Ministry for Research and Technology within the project TASSO under grant no. ITW 8900 C2.

References

[Besnard, 1989] P. Besnard. *An Introduction to Default Logic.* Symbolic Computation — Artifical Intelligence. Springer, 1989.

[Bowen, 1979] K. Bowen. *Model Theory for Modal Logics.* Synthese Library. Reidel, Dordrecht, 1979.

[Brewka, 1991] G. Brewka. Cumulative default logic: In defense of nonmonotonic inference rules. *Artificial Intelligence*, 50(2):183–205, July 1991.

[Delgrande and Jackson, 1991] J. Delgrande and W. Jackson. Default logic revisited. In J. Allen, R. Fikes, and E. Sandewall, eds., *2nd International Conference on the Principles of Knowledge Representation and Reasoning*, pages 118–127. Morgan Kaufmann, 1991.

[Etherington, 1987] D. Etherington. A semantics for default logic. In *International Joint Conference on Artificial Intelligence*, pages 495–498, 1987.

[Łukaszewicz, 1988] W. Łukaszewicz. Considerations on default logic — an alternative approach. *Computational Intelligence*, 4:1–16, 1988.

[Makinson, 1989] D. Makinson. General theory of cumulative inference. In M. Reinfrank, J. de Kleer, M. Ginsberg, and E. Sandewall, eds., *2nd International Workshop on Non-Monotonic Reasoning*, pages 1–18. Springer, 1989.

[Poole, 1989] D. Poole. What the lottery paradox tells us about default reasoning. In R. Brachman, H. Levesque, and R. Reiter, eds., *1st International Conference on the Principles of Knowledge Representation and Reasoning*, pages 333–340. Morgan Kaufmann, 1989.

[Reiter, 1980] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1–2):81–132, 1980.

[Schaub, 1991a] T. Schaub. Assertional default theories: A semantical view. In J. Allen, R. Fikes, and E. Sandewall, eds., *2nd International Conference on the Principles of Knowledge Representation and Reasoning*, pages 496–506. Morgan Kaufmann, 1991.

[Schaub, 1991b] T. Schaub. On commitment and cumulativity in default logics. In R. Kruse, ed., *European Conference on Symbolic and Quantitative Approaches to Uncertainty*, pages 304–309. Springer, 1991.

[Schaub, 1992] T. Schaub. On constrained default theories. Technical Report AIDA–92–2, FG Intellektik, TH Darmstadt, Alexanderstr. 10, D–6100 Darmstadt, 1992.

Fast Solution of Large Interval Constraint Networks

Alexander Reinefeld
Universität Hamburg
Vogt-Kölln-Str. 30
D-2000 Hamburg 54

Peter Ladkin
Universität Bern
Länggasstraße 51
CH-3012 Bern

Abstract

We present a fast solution algorithm for qualitative interval constraint problems that returns solutions to randomly generated problems in less than half a second on the average, with the hardest problems taking only half a minute on a RISC workstation. The fast solution time in these cases is attributed to the extraordinary pruning power of the path-consistency computation, and to the fact that all our randomly generated interval networks of size ≥ 14 were found to be inconsistent.

While inconsistency is relatively easy to prove, our algorithm also solves large *consistent* networks with 100 vertices within reasonable time limits. This is possible, because path-consistency reduces the solution search to an almost linear selection of atomic labels. We conclude that large interval constraint problems can be solved on a serial machine in cubic time in practice, whether the given network is consistent or not.

1 Introduction

Synopsis. In this paper, we present an empirical study on solving qualitative interval constraint networks of the sort first introduced by Allen [All83]. The problem is NP-complete [ViKaVB89]. Our algorithm first employs the relational-matrix composition algorithm of [LadMad88] to check the initial path-consistency and then uses path-consistency further as a pruning technique to reduce the size of the solution space.

In a first set of large-scale experiments, we generated more than 200,000 random networks of various sizes $3 \leq n \leq 20$. Our algorithm solved all problems in an expected time of less than half a second on a SUN SparcStation 1, with the hardest problem taking only half a minute [LadRei91]. The fast solution time is attributed to the following facts:

- The initial path-consistency computation effectively reduces the number of atoms per label, so that the subsequent search for an atomic labeling becomes an almost linear selection.
- Networks of size $n \geq 14$ are inconsistent with more than 99.99% certainty, which is usually detected in the initial path-consistency computation.
- Small networks are usually consistent. But the solution nodes are dense, and so there is little backtracking in the search.

The results were obtained on networks, where the labels are independent, identically distributed random variables drawn from a discrete probability distribution. While such networks are clearly "random" in a mathematical sense, their features may seldom be met in practical applications. In practice, two intervals more likely precede, overlap or contain each other, than that they start or end exactly at the same time point[1]. The exact properties of "realistic" large networks are not known, since to our knowledge there exists no temporal reasoning system that builds *large* consistent networks (– possibly as a consequence of believing that the solution of large networks is infeasible).

For our second set of experiments, we devised a method to synthetically generate large *consistent* networks with dependent interval relations. Our results indicate that networks of 200 vertices can be solved with a reasonable amount of computing time. The maximal problem size is mainly restricted by the available memory space to hold the reduced network matrices in the depth-first search tree. In practice (eg. the technical diagnosis system of [Nök91]), large application problems are unlikely to exhibit the characteristics of random networks, and often domain specific knowledge can be exploited to aid in the solution process. Hence we believe our experiments exhibit behavior worse than can be expected in real applications.

Scope of Experiments and Results. This paper gives answers to the following questions:

- How dense are solutions of randomly generated interval constraint networks?

[1] In planning systems, exact time scales can often be more precise than is needed, in which case Allen's qualitative temporal calculus can suffice for temporal reasoning. For details of the conversion from metric to qualitative reasoning systems (and vice versa) see [KauLad91].

- Is there a difference in path-consistency behaviour between consistent and inconsistent interval networks?
- How good is path-consistency pruning for searching the solution tree?
- How effectively can large consistent networks be solved?
- What size networks can be solved using these methods?

Section 2 briefly introduces the notion of interval constraint networks and presents algorithms for checking path-consistency and for deriving a solution. In Section 3, we summarise the properties of random interval constraint networks of size $3 \leq n \leq 20$ from [LadRei91], and extend our investigation to include also solution density calculations. In Section 4, we concentrate on the solution of large consistent networks. We present methods to speed up the solution search and show statistics obtained in consistent networks of sizes up to $n = 100$. The largest networks we ran were of size 200.

2 Interval Constraint Networks

Interval constraint problems of n intervals can be represented as labelled digraphs, called *networks*, with up to $n \cdot (n-1)/2$ edges. Constraint relations, the labels of the graph, are formally sets (of interval pairs), which are *sums* (set-unions) of atomic constraint relations. The atoms contain interval pairs $\langle\langle x, y\rangle, \langle x', y'\rangle\rangle$, where x, y, x', y' are points on the real line [All83]. We use the seven possible atomic relations between two intervals:

$$\begin{array}{ll}
\textit{Equals:} & 1' = \{\langle\langle x,y\rangle,\langle x',y'\rangle\rangle : x = x' < y = y' \in \mathbf{S}\} \\
\textit{Precedes:} & P = \{\langle\langle x,y\rangle,\langle x',y'\rangle\rangle : x < y < x' < y' \in \mathbf{S}\} \\
\textit{During:} & D = \{\langle\langle x,y\rangle,\langle x',y'\rangle\rangle : x' < x < y < y' \in \mathbf{S}\} \\
\textit{Overlaps:} & O = \{\langle\langle x,y\rangle,\langle x',y'\rangle\rangle : x < x' < y < y' \in \mathbf{S}\} \\
\textit{Meets:} & M = \{\langle\langle x,y\rangle,\langle x',y'\rangle\rangle : x < y = x' < y' \in \mathbf{S}\} \\
\textit{Starts:} & S = \{\langle\langle x,y\rangle,\langle x',y'\rangle\rangle : x = x' < y < y' \in \mathbf{S}\} \\
\textit{Finishes:} & F = \{\langle\langle x,y\rangle,\langle x',y'\rangle\rangle : x' < x < y = y' \in \mathbf{S}\}
\end{array}$$

Taken together with the converse relations $P^\smile$, $D^\smile$, $O^\smile$, $M^\smile$, $S^\smile$ and $F^\smile$, this gives a total of 13 atomic relations, which generate a total of 2^{13} possible different edge labels.

2.1 Path-Consistency

Path-consistency is a necessary condition for consistency but does not imply consistency. Properties of path-consistency were investigated in [Mac77, MohHen86, LadMad88]. The following is summarised from [LadMad88].

Let *intersection* of relations be denoted by '$\cdot$', and *composition* of relations by '$\circ$'. Let P_{ij} be the relational constraint between variables x_i and x_j. Then a path-consistency computation may be regarded as computing the greatest fixed point of the following sets of equations (i.e. the collection of largest possible relations r_{ij} satisfying them):

$$r_{ij} \leq P_{ij}$$

```
function PC (var M: matrix): boolean;
    repeat
        M ← M · M²;
    until M = M²;
    return (M ≠ 0);
end;
```

Figure 1: Path-Consistency Algorithm

$$r_{ij} = r_{ij} \cdot \prod_k (r_{ik} \circ r_{kj})$$

where the symbol $\prod_k$ denotes the product taken over all $k \leq n$. The fixed point is the most-general path-consistent reduction of the original network. This observation leads to the representation of binary constraint networks by $(n \times n)$ matrices of relation symbols M_{ij}, which correspond to the constraints P_{ij}. We call such a matrix a *relational matrix*. By its nature, the matrix has to represent a complete graph, so it contains an entry of **1** (the universal relation, i.e. the union of all 13 atoms) wherever no constraint edge appears in the original problem.

Product and *composition* for relational matrices are defined by the following schemes, which use intersection and composition defined on relations:

$$(M \cdot M')_{ij} = M_{ij} \cdot (M')_{ij}$$

$$(M \circ M')_{ij} = \prod_{k \leq n} M_{ik} \circ (M')_{kj}$$

Define also the power M^n by the usual induction,

$$M^1 = M,$$

$$M^{n+1} = (M^n) \circ M$$

So, for example, $M^2 = (M \circ M)$, as we use in our path-consistency algorithm in Figure 1. Algorithms in the literature differ mainly in the details of the iterations over the triangles. Note that our PC-function reduces the relational matrix M by side effect. If any zero edge is detected in M, PC returns false, indicating that the relation matrix M is not path-consistent.

We say that a network *fails* if a path-consistency computation detects inconsistency, and that it *succeeds* if it stabilises without detecting inconsistency. Succeeding networks are usually consistent, as shown in Figure 3, but this can not be always the case since the problem is NP-complete and path-consistency is cubic time.

2.2 Consistency

A network is *consistent*, if there exists a consistent atomic labeling. Our solution algorithm works as follows:

> *Perform a path-consistency computation until stability. If the network succeeds, randomly select one edge and an atom from the label on that edge. Then run again the path-consistency algorithm and fix an atom on another edge until all labels are atomic. If the network does not succeed at some point, backtrack and choose another atom on that label.*

```
function CC (var M: matrix; i, j: integer): boolean;
{Consistency Computation, starting with edge M_ij}
    M' ← M;                    {save matrix M}
    for each atom a_k ∈ M_ij do begin
        M_ij ← a_k;
        if PC(M) then
            if i, j are last edges
            or CC(M, next_i, next_j) then
                return (true);
        M ← M';                {restore M}
    end;
    return (false);            {no consistent labeling}
end;

begin {main}
    if PC(M) then
        if CC(M, 1, 2)
            then writeln ("consistent");
            else writeln ("inconsistent");
    else writeln ("not path-consistent");
end.
```

Figure 2: Consistency Computation Algorithm

Figure 2 describes the solution algorithm in a Pascal-like pseudo code. The main routine first invokes PC (Figure 1) to check the path-consistency of the given network matrix M. If the network succeeds, the consistency computation CC is started. CC fixes an arbitrary atomic label a_k on the first edge M_{12}. If the reduced network M is still path-consistent, CC recursively deepens on the next edge, where it again selects an arbitrary atomic label. When all selected atomic labels form a path-consistent network, the whole network is consistent [LadMad88].

Note that each PC-computation reduces the number of labels in the network matrix M as a side effect. Hence the size of the search space constantly reduces. The resulting search tree is bushy near the root and gets thinner in deeper levels. In practice, the solution process turns out to be essentially a linear selection of atoms. Very little backtracking occurs, because path-consistency is a very effective pruning technique.

3 Results for Random Networks

In a first set of experiments, we generated interval constraint networks where the labels are independent, identically distributed random variables drawn from a discrete probability distribution. The networks have the following properties:

- Each atom within a label occurs with the same probability.
- Each label within the network occurs with the same probability.

We excluded networks with zero-labels, since they are known to be inconsistent. We generated 10,000 networks of sizes 3 to 20, and also 10,000 networks with pointisable labels[2] for the same range of sizes. Figure 3 shows the percentage of networks of each size that succeeded in the first path-consistency computation, and also those that were later found to be consistent by the solution search. For pointisable networks, path-consistency implies consistency [LadMad88], hence Figure 3 depicts only one graph with pointisable networks.

The graphs show a sharp jump in failure rate from networks of size 8 (roughly 20% fail) to networks of size 11 (roughly 80% fail). The networks in this range (we call it the *transition range*) took particularly long to search, in comparison with networks outside the range. Even so, the average solution time required within the transition range was less than half a second of CPU time on a Sun SparcStation 1. Amongst the 200,000 problems generated, we saw no case that needed longer than half a minute to solve.

Moreover, we found that *all* general networks that we randomly generated with more than 13 variables are inconsistent, as are all pointisable networks with more than 11 variables. While at sizes 14 to 17, some small number ($\leq 1\%$) of these inconsistent networks succeed, inconsistency of large networks is usually detected right in the initial path-consistency computation. But even in the rare cases when a large network succeeds, the initial path-consistency computation has not been done in vain. It then serves to reduce the size of the labels (and therefore the branching degree of the search tree), which speeds up the subsequent solution search.

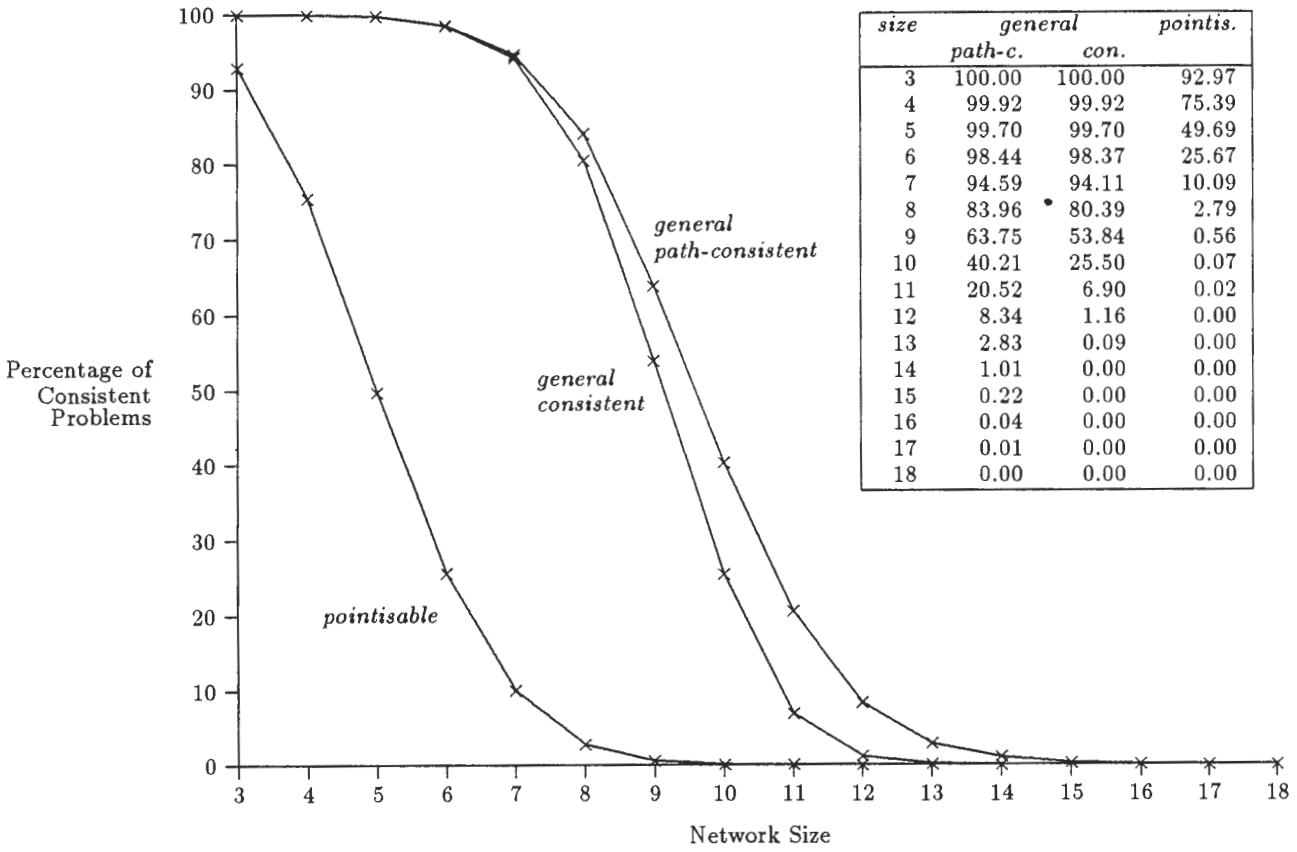
3.1 Time to Solve

The CPU time needed to solve a randomly generated interval constraint network depends on the size of the network, the average label size, and the number of solutions in the search space.

Network Size. The larger the network, the less likely it is to be consistent, see Figure 3. All 70,000 random networks of size > 13 are inconsistent. Moreover, inconsistency is easy to prove, since only one zero-label must be found. Zero-labels are usually detected early in the initial path-consistency computation, so that no search is required.

Our detailed statistical results (not shown here) indicate, that the large consistent networks need less iterations to stabilise in the path-consistency computation than the consistent ones. Inconsistent networks of size $n > 14$ usually stabilise in the first iteration of the initial path-consistency computation. The two bottom graphs in Figure 4 give the number of iterations averaged over all (consistent and inconsistent) problems. In the worst case, at the size 9-networks of the random general networks, an average of 3.3 iterations are needed to stabilise. On both sides, the graph quickly levels off to less than 2 iterations. Only the large consistent networks of size

[2] Pointisable networks are those in which every 2-subnetwork may be represented by a point network, and therefore the whole network may be represented by a single, larger, point network. They were defined in [LadMad88], and in [vBeCoh89]. As a practical application, Nökel's technical diagnosis system [Nök91] uses a subclass of pointisable networks.



size	general		pointis.
	path-c.	con.	
3	100.00	100.00	92.97
4	99.92	99.92	75.39
5	99.70	99.70	49.69
6	98.44	98.37	25.67
7	94.59	94.11	10.09
8	83.96	80.39	2.79
9	63.75	53.84	0.56
10	40.21	25.50	0.07
11	20.52	6.90	0.02
12	8.34	1.16	0.00
13	2.83	0.09	0.00
14	1.01	0.00	0.00
15	0.22	0.00	0.00
16	0.04	0.00	0.00
17	0.01	0.00	0.00
18	0.00	0.00	0.00

Figure 3: Percentage of Consistent Networks, 10,000 Problems

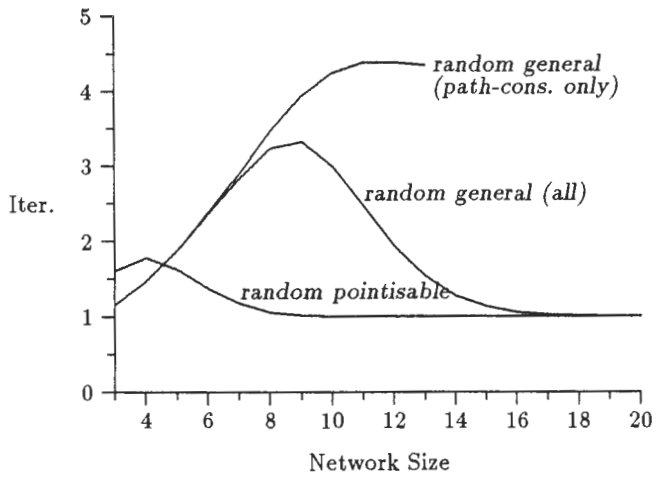


Figure 4: Iterations in Initial Path-Consistency Computation, 10,000 Problems

> 7 (shown in the top graph) need more always than 4 iterations to stabilize.

Average Label Size. Figure 5 illustrates the pruning power of the path-consistency computation. Here, the average number of atoms per label is plotted for networks of size 3 to 20. The 10,000 randomly generated problem instances are split into two groups: the ones succeeding in the initial path-consistency computation (see the two graphs at the top), and the ones that fail (the bottom graph). We plotted only data points that are supported by a sufficiently large number of samples (well above 100 problem instances).

Initially, the average number of atoms per label is $13/2 = 6.5$ for all network sizes, see the straight line at the top. For the failed networks (bottom graph), the first iteration of the path-consistency computation almost halves the label sizes, especially in the very small and the very large networks. The following iterations then reduce the remaining atoms further, until a zero-label is found, indicating that the network is inconsistent. Note that path-consistency is least effective in achieving reductions in networks of sizes 9 to 11. These are also the networks where the most iterations are needed to stabilise (see Figure 4).

For succeeding networks the reduction is not as big. But again, half of the work is done with the first iteration of the initial path-consistency computation. Maximal reductions are achieved in networks of size 10.

In general, one can predict right after the first iteration of the path-consistency computation whether a network will eventually succeed or not: If the first iteration leaves more than an average of 5 atoms per label, the network will very likely succeed, otherwise it will probably fail. The discrimination is more dramatic in the very small and very large networks than in those in the transition range.

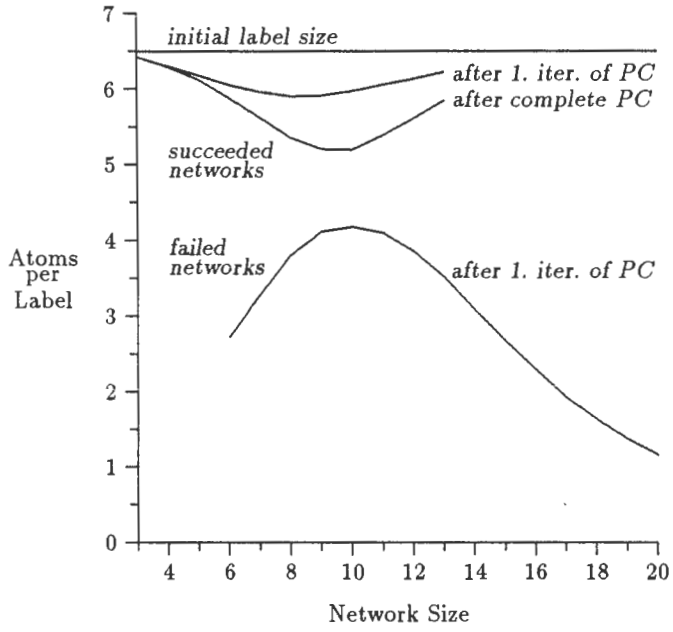


Figure 5: Average Label Sizes, 10,000 Problems

Solution Density. For succeeding networks, the initial path-consistency computation leaves a number of atoms per label, which must be further reduced by means of search until a consistent atomic labeling is found (if there is one). The effort needed to solve the problem depends on the density of the solution nodes in the search space. We investigated how many of the initial atoms (after the initial path-conistency computation) are actually part of a solution. This gives us a measure of the pruning power of the path-consistency computation.

For this purpose, we modified our algorithm to examine the whole search space and return *all* solutions, rather than only one. We used the method described above to generate as many random problems as needed to obtain exactly 100 consistent networks of each size. All data is averaged over these 100 consistent networks.

	After Initial *PC*		Consistency Search			
n	b	b^n	$s(n)$	$\frac{b^n}{s(n)}$	I	$\frac{I}{s(n)}$
3	6.33	254	10	25	22	2.20
4	6.17	1,149	114	13	262	2.30
5	6.00	7,776	744	10	1,817	2.44
6	5.87	40,910	4,272	10	10,756	2.52
7	5.62	177,074	10,446	17	26,636	2.55
8	5.46	789,844	14,814	53	38,967	2.63
9	5.31	3,356,222	16,899	198	44,780	2.65
10	4.91	8,143,572	87,730	93	230,703	2.63

Table 1: Solution Density in 100 Consistent Problems

Table 1 shows the results of the networks of sizes $3 \leq n \leq 10$. The second column shows the branching factor of the brute force search tree b. It is given by the average number of atoms per label after the initial path-consistency computation. The 3rd column, b^n, is the number of bottom positions in the search tree, equivalently, the number of paths through the tree, when *no* pruning technique (like path-consistency) is used. It includes inconsistent as well as all consistent paths in the original constraint problem. Hence, b^n is the size of the search space of a brute force search. Dividing b^n by the average solution number, $s(n)$, gives the proportion of paths to solution paths of a simple brute force search without path-consistency pruning (see 5th column)[3]. For network sizes $n < 8$ this number lies between 10 and 25, that is, every 10th to 25th leaf node (path) is a solution. In the larger networks, more nodes must be searched to find a solution, with the peak lying at size 9-networks. But even in this worst case the solutions are quite dense, considering the enormous size of the search space. We conjecture, that in some cases a simple brute force search strategy (possibly mixed with occasional consistency checks on every i-th tree level) should suffice to quickly obtain a solution.

When path-consistency pruning is applied in every interior node of the search tree, even fewer nodes are expanded. The second-to-last column shows the total number of (interior) nodes I visited in the search process. Dividing I by the total number of solutions, $s(n)$, gives the reciprocal of the *pruned solution density*, i.e. when path-consistency is used throughout the search. On the whole range of network sizes, this number lies between 2.2 and 2.7, that is, the probability that a path is a solution path is 0.4. This constancy over the whole range of network sizes indicates that the pruned solution search is likely to be equally effective for all sizes of network[4].

From this it follows that the path-consistency computation dominates other aspects of the search algorithm as the network size increases, and there are an expected constant number of path-consistency computations per problem. It further follows from [Mac77, MacFre85] that we can expect to obtain solutions in practice in *cubic time* for serial computations.

4 Large Consistent Networks

The larger the network, the more critical is the order of node expansions in the solution process. On one hand, path-consistency requires cubic time, which may be impractical for solving large problems. On the other hand, the program might run out of storage space, because the solution process builds a search tree of depth $d = n \cdot (n-1)/2$, and even a simple depth-first search needs d^2 words to store the intermediate (possibly reduced) networks on a stack. For size 100-networks, a maximum of 24 million words are needed. In this section, we investigate these two features of large-problem solving.

We first describe methods to speed up the path-consistency computation and then enhancements to the

[3] The reciprocal of this number is the *brute-force solution density*. We find the results tabulated in this form easier to read.

[4] But one cannot conclude from this that the density of solutions at *each interior search node* is constant. In fact, we found that the structure of the search tree is highly unbalanced, with solution nodes clustering in certain subtrees.

solution search. At the end of this Section, we present results obtained in large consistent networks of sizes up to $n = 100$.

4.1 Complexity of Path-Consistency

Path-consistency computations are known to be serial cubic time [Mac77], or parallel $O(n^2 \log n)$ time, and iterative algorithms take $\theta(n^2)$ time [LadMad88]. A complete network of size n has $n \cdot (n-1)/2$ edges. For every edge, the compositions of all $n-2$ alternative length-2-paths (=triangles) must be intersected with the original edge, giving a total of

$$\frac{n \cdot (n-1) \cdot (n-2)}{2}$$

compositions and intersections. Computing the converse relations (the other triangular matrix) requires another $n \cdot (n-1)/2$ operations. For size 10 networks, only 405 operations are needed, but for size 50, we get 60,025 and for size 100 even 490,050 operations. To make things worse, path-consistency is an iterative process, that usually needs more than one iteration to stabilize. While in theory, a maximum of

$$13 \cdot \frac{(n-1) \cdot (n-2)}{2}$$

iterations are required [All83], path-consistency stabilises much faster in praxis. Large random networks stabilise earlier than small ones, as is evident from Figure 4. This convenient property is due to the larger number of intersections, which effectively reduces the size of the labels.

4.2 Improved Path-Consistency Algorithm

The path-consistency algorithm of Section 2.1 simply recomputes all labels in every iteration. In practice, it suffices to recompute only the triangles of edges P_{ij} whose labels changed in the previous iteration. Some authors [Mac77,All83] proposed a queue data structure for maintaining the triangles that must be recomputed. The path-consistency computation then simply proceeds until the queue is empty. For our implementation, it turned out to be faster to flip a bit in the network matrix, denoting that this element must be recomputed next time. This has the advantage that the evaluation order will be preserved in the next iteration.

4.3 Improved Solution Algorithm

We improved the solution process of our implementation along the same lines. Each time the consistency computation function, CC, selects a certain atom on an edge P_{ij}, the bits P_{ik} and P_{kj} for $1 \leq k \leq n$ are flipped to tell the path-consistency procedure PC that only these edges must be recomputed. This saves computing time, especially in the large networks.

Our program spends roughly 98% of the total CPU time in path-consistency computations, with more than 3/4 being spent in path-consistency checks invoked by the solution search. In the following, we present methods to speed up the solution algorithm by reducing the number of path-consistency computations.

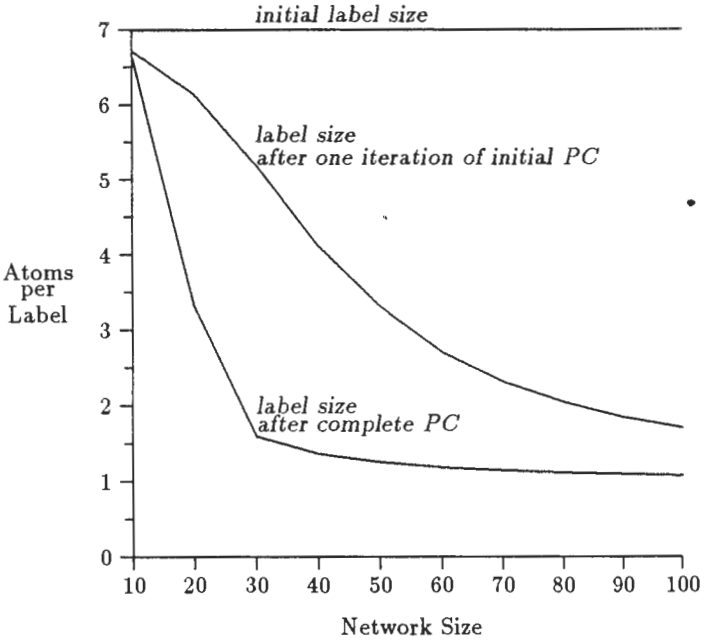


Figure 6: Label Sizes in Large Consistent Networks (1000 Problems)

Skip atomic edges. In the search process, skip all edges with atomic labels. Due to the previous path-consistency computation, atomic labels are known to be consistent to the other labels and hence do not further restrict the search space. This does not only save computing time, but – more important – it saves memory space. For the network sizes we investigated, skipping atomic edges turns out to be absolutely necessary.

Select 1′-atoms first. If an edge of the network contains an *equals*-relation, reduce the dimension of the network matrix by joining the two vertices and intersecting the labels of all adjacent edges. Formally: Let P_{ij} contain a $1'$ (possibly among other atoms). Then select the $1'$-relation first and do for all $k \leq n : P_{ik} \leftarrow P_{ik} \cap P_{jk}$ and $P_{jk} \leftarrow P_{jk} \cap P_{ik}$.

If $1'$ is the only atom of P_{ij}, delete the i-th row and the i-th column from the network matrix. The label intersection has already been done in the preceeding path-consistency computation.

Select point-meeting-relations first. Select labels first, that have only few entries in the composition table. This applies for intervals starting or ending at exactly the same time instance: *equals, meets, starts, finishes* and their converses. The motivation is, to cut the search space by reducing the amount of atoms remaining in the network.

4.4 Empirical Results on Large Consistent Networks

With the random-generation method we did not obtain any consistent network of size $n \geq 14$. All 70,000 networks of size 14 to 20 were found to be inconsistent. This is because in the larger networks, more label compositions are intersected, which reduces the number of pos-

sible relations between two intervals. Hence, the probability of two relations being consistent is low.

Since we do not know of any method to generate truly random large consistent networks, we generated consistent networks with *dependent* interval relations in the following way:

> *Pick a number of intervals lying on a metric scale $t_1, \ldots, t_m$ of discrete time instances and compute the relations that hold between each interval pair. This gives a set of* initial relations *that, taken together, form an* initial solution. *Then produce a final network by taking the sum of the initial relations and some other randomly chosen labels.*

While the generated networks are clearly not random, they are consistent with at least one solution: the initial solution. The larger the time scale, the less likely it is for two intervals to have one or two end points in common. In our case – we chose a scale of 100 discrete time instances – the labels *before, overlaps, during* and their converses occur with 16% probability, while *meets, starts* and *finishes* have only a 0.6% probability. We generated a total of 1000 consistent networks for sizes between 10 and 100. Figure 6 shows our experimental results: The topmost graph shows the initial label sizes, the middle graph shows the label sizes after the first iteration of the initial path-computation, and the bottom graph shows the label sizes at the end of the first path-consistency computation. Similar to the results obtained in random networks (Figure 5), these graphs show that most of the work is done after the very first iteration of the initial path-consistency computation. When the initial path-consistency computation is completely finished, even less atoms per label remain. In networks of size 80, eg., an average of only 1.1 atoms per label remain to be examined in the solution search.

Our detailed statistical data (not presented here) indicates that the larger networks need fewer iterations to stabilise. The peak (of about 7 iterations) is reached at the size 20-networks. In the larger networks, about 4 iterations are required. Path-consistency is more effective for pruning in the larger networks.

5 Conclusions

According to our experience, large interval constraint networks can be solved with a reasonable amount of computing resources, no matter whether they are consistent or not. The initial path-consistency computation effectively eliminates inconsistent atomic relations from the network. Inconsistency is usually detected in the initial path-consistency computation. For the consistent networks, most of the atoms that remain after the first path-consistency computation are part of a solution. Very little backtracking is necessary in the solution search.

The largest consistent networks we checked were of size 200. Our experiments were limited by the main memory space of our workstation (8 Mbytes). Our solution algorithm stores the whole relation matrix in every interior node of the search tree. In the worst case, $[n\cdot(n-1)/2]^2$ memory words are needed. This follows because the search tree has depth $n\cdot(n-1)/2$, and for each level one a triangular relation matrix of size $n\cdot(n-1)/2$ is stored. If some labels in the network are already atomic (which is usually the case), less space is needed.

Solution time is dominated by the time needed to check the path-consistency, an expected constant number of times. Solution is thus serial *expected cubic time*, even for large networks. We conclude that the search for better search techniques can be halted. Instead it seems more important to speed up the path-consistency computation. Our future research therefore includes implementation of effective parallel path-consistency computation schemes on a Transputer network.

References

All83: Allen, J.F., *Maintaining Knowledge about Temporal Intervals*, Comm. ACM 26 (11), November 1983, 832-843.

DeMePe91: Dechter, R., Meiri, I., and Pearl, J., *Temporal Constraint Networks.* Artificial Intelligence 49, 1991, 61-95.

KauLad91: Kautz, H.A. and Ladkin, P.B., *Integrating Metric and Qualitative Temporal Reasoning*, Proceedings of AAAI-91, the 9th National Conference on AI, AAAI Press 1991.

LadMad88: Ladkin, P.B., and Maddux, R.D., *On Binary Constraint Networks*, Kestrel Institute Technical Report KES.U.88.8, under revision for publication.

Lad90: Ladkin, P.B., *Constraint Reasoning With Intervals: A Tutorial, Survey, and Bibliography*, ICSI Report TR-90-059, Sept. 1990.

LadRei91: Ladkin, P.B., and Reinefeld, A., *How to Solve Interval Constraint Networks: The Definitive Answer – Probably*, ICSI Report TR-91-063, Nov. 1991.

Mac77: Mackworth, A.K., *Consistency in Networks of Relations*, Artificial Intelligence 8, 1977, 99-118.

MacFre85: Mackworth, A.K., and Freuder, E.C., *The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems*, Artificial Intelligence 25, 65-74, 1985.

MohHen86: Mohr, R., and Henderson, T.C., *Arc and Path Consistency Revisited*, Artificial Intelligence 28, 1986, 225-233.

Nök91: Nökel, K., *Temporally Distributed Symptoms in Technical Diagnosis*, Lecture Notes in Artificial Intelligence 517, Springer Verlag 1991.

vBeCoh89: van Beek, P.G., and Cohen, R., *Approximation Algorithms for Temporal Reasoning*, in Proceedings of IJCAI89, the 11th Joint Conference on Artifical Intelligence, 1291-1296, Morgan Kaufmann 1989.

ViKaVB89: Vilain, M., Kautz, H., and van Beek, P.G., *Constraint Propagation Algorithms for Temporal Reasoning*, in Weld and de Kleer, eds., *Readings in Qualititative Reasoning About Physical Systems*, Morgan Kaufmann 1989.

ORDERING HEURISTICS FOR ARC CONSISTENCY ALGORITHMS*

Richard J. Wallace
Eugene C. Freuder
Computer Science Department
University of New Hampshire
Durham, NH 03824 USA

Abstract

Arc consistency algorithms are used in solving constraint satisfaction problems and are important in constraint logic programming languages. Search order heuristics for arc consistency algorithms significantly enhance the efficiency of their implementation. In this paper we propose and evaluate several ordering heuristics. Care is taken with experimental design, involving random problems, and statistical evaluation of results. A heuristic is identified which yields about 50% savings on average, using the standard measure of consistency pair checks, with reasonable heuristic computation cost.

1 Introduction

Arc consistency insures that any two mutually constraining problem variables are mutually consistent: given a value for one, we can find a value for the other which satisfies the constraint between them. The constraint specifies which pairs of values can be simultaneously assumed by the pair of variables.

Arc consistency is a fundamental concept in constraint-based reasoning [Mackworth, 1987] and has played a significant role in constraint logic programming [Dincbas *et al.*, 1988; Van Hentenryck, 1989]. Arc consistency can also be used to answer temporal reasoning questions [Meiri, 1992]. Various forms of arc consistency have been utilized in various roles to solve constraint satisfaction problems (CSPs), i.e. to find values, for a set of variables, that satisfy a set of constraints. Many artificial intelligence problems, from scene analysis to scheduling, have been viewed as CSPs.

For some classes of problems arc consistency processing alone essentially finds a solution [Freuder, 1982; Deville and Van Hentenryck, 1991]. Some algorithms repeatedly employ generalized forms of arc consistency to find solutions to arbitrary problems [Mackworth, 1977a; Freuder, 1978]. Arc consistency can be used for "preprocessing" before further search, and may simplify a problem to the point where little if any subsequent search effort is needed, especially wheninitial "boundary conditions" are known [Waltz, 1975]. Full or partial arc consistency can be combined with backtrack search in "hybrid" algorithms [Nadel, 1989].

While there has been some indication in the literature that ordering heuristics can improve the performance of the *relaxation* (*constraint propagation*) algorithms that achieve arc consistency [Waltz, 1975], these heuristics have not received systematic study. (This is to be contrasted with the considerable attention devoted to ordering heuristics for constraint satisfaction backtrack search.)

This paper initiates such a systematic study. We identify factors that determine the efficiency of constraint propagation. Probabilistic arguments suggest the relationship between these factors and certain easily measurable problem characteristics. Several ordering heuristics are proposed based on these problem characteristics. Careful testing on random problems verifies our expectations of increased efficiency.

Our best heuristics halve the number of constraint checks (a standard measure of performance). The worst-case complexity of the overhead involved in computing the ordering is linear in the number of constraints, and does not add to the overall worst-case complexity of the arc consistency algorithm. Computation time data confirms the efficacy of the heuristic.

Most of our experiments are based on the standard AC-3 arc consistency algorithm [Mackworth, 1977b]. AC-4 [Mohr and Henderson, 1986] has a better worst-case bound than AC-3 (indeed its worst-case bound is optimal), but incurs the time/space costs of building and maintaining more elaborate data structures. AC-4 is also subject to ordering improvement, and we demonstrate this experimentally as well; however, in our tests AC-4 was very much less efficient than AC-3.

We would expect ordering heuristics to improve some partial arc consistency algorithms [Haralick and Elliott, 1980; Dechter and Pearl, 1988; Nadel, 1989; Freuder and Wallace, 1991]. The ideas should be generalizable to higher level consistency algorithms [Montanari, 1974; Mackworth. 1977b; Freuder, 1978; Cooper, 1989].

*This work was supported by the National Science Foundation under Grant No. IRI-8913040. The government has certain rights to this material. Some of this work was done while the second author was a Visiting Scientist at the MIT Artificial Intelligence Laboratory.

Section 2 reviews basic concepts. Section 3 discusses factors that determine efficiency of relaxation for any problem and Section 4 discusses why some orderings would be expected to enhance performance *a priori*. Sections 5 and 6 describe methods and results, respectively, of experiments with random constraint satisfaction problems designed to assess the effectiveness of orderings based on these heuristics. Section 7 considers the effort required to obtain good orderings with specific heuristics. Section 8 summarizes the conclusions.

2 Basic Concepts

A constraint satisfaction problem (CSP) involves a set of n variables, v_i, each having a *domain* of values, d_i that it can assume. In addition, the problem is subject to some number of *binary constraints*, C_{ij}, each of which is a subset of the Cartesian product of two domains, d_i X d_j. A binary constraint specifies which pairs of values can be simultaneously assumed by the pair of variables. (Only binary constraints are considered here; higher-order CSPs can also be represented by binary CSPs [Rossi *et al.*, 1989].) A CSP is associated with a *constraint graph*, where nodes represent variables and arcs represent constraints (Figure 2).

AC-3 involves a sequence of tests between pairs of constrained variables, v_i and v_j: we say that v_i is relaxed against v_j. Specifically, values in v_i are checked against the constraint between v_i and v_j to see if they are supported, i.e. are consistent with at least one value in the domain of v_j; unsupported values are deleted. The AC-3 algorithm is shown in Figure 1. All ordered pairs of constrained variables are first put in list L. Each pair, (v_i, v_j), is removed and v_i is relaxed against v_j. When values are deleted, pairs may need to be added to L to determine if these deletions lead to further deletions.

```
Initialize L to {(vi, vj) |
        a constraint exists between vi and vj}.
While L is not empty
        Select and remove (vi, vj) from L.
        Relax vi against vj.
        If relaxation removes any values from vi,
            add to L any pairs (vk, vi), kπj, such that
            there is a constraint between vk and vi and
            (vk, vi) is not already present in L.
```

Figure 1. The AC-3 algorithm.

The order in which pairs are selected for relaxation reduces to a question of how L is ordered, where the first pair in L is always selected. Specifically, how should L be ordered in the initialization phase and how should we insert additions to L during the relaxation phase? Ordinarily the initialization order is simply derived from the order in which the problem is described, and L is maintained as a queue [Mackworth and Freuder, 1985; Nadel, 1989]. We are concerned with heuristics for ordering L which result in more efficient processing.

3 Factors That Determine Efficiency of Relaxation

Differences in the efficiency of relaxation that depend on ordering can be ascribed to two factors: (i) if a value is deleted from d_i when v_i is relaxed against v_j, less work is performed if other constraints that include v_i are tested after this restriction rather than before (minimizing the number of tests before deletion), (ii) work is reduced if pairs with nodes adjacent to the node relaxed are already on the list (minimizing the number of list additions, or relaxations). Both factors are consistent with the principle that, in good orderings, domain values are removed as quickly as possible (the "ASAP principle"). This idea was first stated by [Waltz, 1975] in his discussion of filtering descriptions of line drawings, *viz*, "The basic heuristic for speeding up the program is to eliminate as many possibilities as early as possible" (p. 60). However, it is not so much a heuristic as a characteristic of efficient performance, since conformity to this principle cannot be determined in advance.

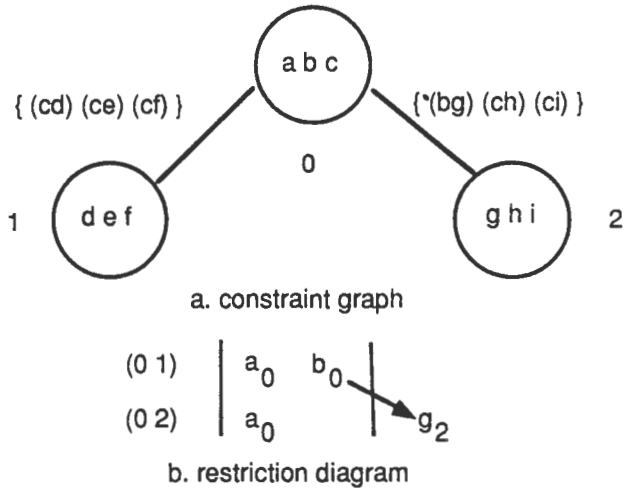


Figure 2. CSP illustrating relaxation principles.

One aspect of CSP structure that can affect efficiency via the order of relaxation is the presence of sequential dependencies in the pattern of domain support. For example, suppose the values within domain d_j that support a value in d_i are not supported by values in another domain, d_k. In this case, relaxing v_j against v_k removes the values in d_j that supported the values in d_i, and, if v_i is then relaxed against v_j, the latter values are removed the first time this variable pair is tested. On the other hand, if v_i is relaxed against v_j before the latter is relaxed against v_k, these values cannot be removed, and more values are retained that will eventually be eliminated. Since failure to observe such dependencies will result in pairs being put back on the list after they have been examined, this is a special case of minimizing additions to the list of variable pairs. (A specific reference to sequential dependencies is made in [Gaschnig, 1974].)

These principles are illustrated with a simple example (Figures 2-3). The CSP in this case has three variables and two constraints, so its constraint graph is a tree. Domains have the same number of values and constraints the same number of acceptable pairs. Figure 2 also includes a *restriction diagram*, showing all values that could possibly be deleted by relaxation and any dependencies between them. Variable pairs with at least one domain subject to restriction are shown on the left, and each column to the right includes all values with the same depth of dependency (subscripted with variable names). The diagram, therefore, gives a 'parallel view' of relaxation. Dependency relations between specific values are shown by arrows.

start			(abc)	(def)	(ghi)
good ordering (conforms to ASAP rule)					
0 ag 1	remove a,b	7 cks	(c)	(def)	(ghi)
2 ag 0	remove g	3	(c)	(def)	(hi)
0 ag 2	---	1			
1 ag 0	---	3			
fails to minimize pair checks (no list additions)					
1 ag 0	---	9 cks			
0 ag 1	remove a,b	7	(c)	(def)	(ghi)
2 ag 0	remove g	3	(c)	(def)	(hi)
0 ag 2	---	1			
fails to minimize additions (no sequent depend. violations)					
1 ag 0	---	9 cks			
0 ag 2	remove a	6	(bc)	(def)	(ghi)
0 ag 1	remove b	4	(c)	(def)	(ghi)
2 ag 0	remove g	3	(c)	(def)	(hi)
1 ag 0	---	3			
fails to respect sequential dependency					
2 ag 0	---	8 cks			
0 ag 1	remove a,b	7	(c)	(def)	(ghi)
0 ag 2	---	2			
1 ag 0	---	3			
2 ag 0	remove g	3	(c)	(def)	(hi)

Figure 3. Relaxation with different orderings.

The course of relaxation of this problem for different AC-3 list orderings is shown in Figure 3. Constraint checks are shown on successive rows, including the variable relaxed (*i* ag *j* indicates that v_i is relaxed against v_j), the values (of d_i) removed, the number of value pairs that were checked (assuming a lexicographic order), and the current domains after each instance of domain restriction. The first example shows an ordering consistent with the ASAP principle: when v_0 is relaxed against v_1, the pair, (0 2), which includes an adjacent node is still on the list, the dependency between b in d_0 and g in d_2 is respected, and v_0 is relaxed against v_2 and v_1 against v_0 after their domains have been reduced. In the second example v_1 is relaxed against v_0 at the beginning, when their domains have their original sizes, so more pair checks are required. In the third example, v_0 is relaxed against v_2 when the pair (1 0) has already been removed from the list; since a value is deleted from the domain of v_0, (1 0) must be put back on. In the fourth example, the sequential dependency between b in d_0 and g in d_2 is violated; nothing is deleted from d_2 when v_2 is first relaxed against v_0, and the lack of support for g in the former domain must be discovered in a second test.

4 Rationale for Order Heuristics

Many ordering heuristics can be devised, based on three major features of constraint satisfaction problems: (i) the number of acceptable pairs in each constraint (the constraint size or *satisfiability* [Nadel, 1988]), (ii) the number of values in each domain, (iii) the number of binary constraints that each variable participates in, equal to the degree of the node of that variable in the constraint graph. Simple examples of heuristics that might be expected to improve the efficiency of relaxation are: (i) ordering the list of variable pairs by increaing relative satisfiability, i.e. the proportion of possible pairs that are acceptable, (ii) ordering by increasing size of the domain of the variable relaxed *against*, (iii) ordering by descending degree of node of the variable relaxed.

However, it is important to note that the principles of efficient relaxation described in Section 3 do not depend on differences in such features. Thus, in the example given there, domain size and satisfiability are both constant. (It is easy to construct similar examples where degree of node does not vary.) Conversely, variation in these features does not guarantee that a particular heuristic will enhance efficiency. For example, a smaller domain size does not itself imply that relaxation is more likely, since the acceptable pairs can include all domain values. This means that the case for ordering heuristics depends on probabilistic arguments. In other words, we must show that a given ordering can be expected to enhance efficiency, if we can make some assumptions about expected failure of domain support.

For any binary constraint, the range of possible satisfiabilities, from 0 to the product of the participating domains, can be divided into three parts, in which relaxation is inevitable, possible, or impossible, respectively. These subranges are:

(inevitable)	0 to $d_{max} - 1$
(possible)	d_{max} to $[(d_{max} * d_{min}) - d_{min}]$
(impossible)	$[(d_{max} * d_{min}) - d_{min} + 1]$ to $d_{max} * d_{min}$

The basis for this division is not hard to see: (i) if the constraint size is less than the size of the largest domain, not all values can be supported, (ii) if at least one value of the larger domain is not supported, the largest possible constraint size cannot be larger than the product of the smaller domain size and one less than the larger one. If v_i is relaxed against v_j, an alternative description of these intervals is:

(inevitable)	0 to $d_i - 1$
(possible)	d_i to $[(d_i * d_j) - d_j]$
(impossible)	$[(d_i * d_j) - d_j + 1]$ to $d_i * d_j$

In accordance with the ASAP rule, an ideal ordering heuristic would insure that any pairs for which relaxation is more likely would be put at the beginning of the list. In particular, those pairs for which relaxation is inevitable

would be tested first. Any reordering after a domain restriction should also have this property.

Since, for any domain pair, relaxation is at least as likely if the satisfiability is smaller, ordering based on increasing satisfiability is an obvious candidate heuristic. Two variants on this idea are ordering by relative satisfiability and by satisfiability *per se*, which does not discount the effects of domain size. A third heuristic of this type is suggested by the following argument. Suppose that associated with each pair of values is a probability, p, that it will be included in the set of acceptable pairs. Then the following relation holds between the sizes of the domains, d_i and d_j and the expected size of the set of constraint pairs based on these domains, $|C_{ij}|$:

$$|C_{ij}| = |d_i| * |d_j| * p.$$

In this case, if v_i is to be relaxed against v_j, some domain restriction would be expected on average when $|d_j| * p$ is less than 1, because in this case the expected size of the constraint set is smaller than the domain size. Hence, ordering by $|d_j| * p$, or equivalently, by $|C_{ij}| / |d_i|$, may be an effective heuristic.

Using the partitioning of satisfiabilities, an argument can also be made for the potential usefulness of heuristics based on domain size. Suppose that domains of size k are to be relaxed against domains of size 1, 2, and 4. For these cases, the relative number of possible values for the satisfiability that are in the "inevitable" range is k-1/k, k-1/2k or k-1/4k, respectively; in general, this number is approximately $1/|d_j|$. Similarly, the relative numbers of possible satisfiability values within the possible and impossible ranges are approximately $1 - 1/|d_j| - 1/|d_i|$ and $1/|d_i|$, respectively. Thus, with larger $|d_j|$, the relative number of values in the impossible range is smaller, and, by the symmetry of the binomial expansion, so is the relative number of possible constraints that fall within this range. This suggests that ordering by increasing $|d_j|$ can be an effective heuristic. Another possibility is to order the pairs by decreasing difference in size between the domain of the variable relaxed and the domain of the variable relaxed against, with signs of the differences considered.

With other factors equal, if a variable is associated with more constraints, it is more likely that a given value in its domain will not be supported in one of these constraints. This suggests that variables with more constraints should be tested first, i.e. that ordering by decreasing degree of the node in the constraint graph of the problem would be an effective heuristic.

5 An Example

The effect of some of these heuristics is illustrated with a variation of the four queens problem. In the basic problem, four queens must be placed on a 4 X 4 board so that no two can attack each other. One way to represent this as a CSP is to define the rows, from top to bottom, as variables and the columns, from left to right, as domain values. In the present variant, the first domain has only one value, associated with a queen in column 2. Relaxation reduces this problem to the solution, ((2) (4) (1) (3)), i.e., column 2 for variable 1, column 4 for of variable 2, etc.

Table 1 shows the number of additions to the list and the total number of value pairs checked, for some of the heuristics introduced in the last section. (Variable pairs that are equivalent under a heuristic are placed on the list in lexicographic order, e.g., (1 2), (2, 4), (3, 1).). Results for two reference orderings are also included: (i) the lexicographic ordering maintained as a queue was used explicitly by [Nadel 89], while queue ordering appears to be the method generally used, as indicated above, (ii) an alternative baseline is provided by the mean of ten random orderings (whose derivation is described below). Results for reversed orderings are also introduced here; this is a more sensitive method for detecting the influence of a heuristic, even when it does not yield results that are appreciably different from the reference orders.

For this problem, the effect of ordering is apparent and sometimes dramatic. Orderings that might be expected to yield good performance, viz, increasing satisfiability and increasing size of the domain relaxed against, yield lower values for the measures of work in comparison with the standards. By the same token, an excessive amount of work is required for relaxation when the ordering reverses one of the 'good' orders. It should also be noted that treating the list as a stack rather than a queue does not improve efficiency. This was the typical finding for all problems, when these or other, more elaborate, placement strategies were tested.

Table 1
Efficiency of AC-3 with Different Ordering Heuristics
(Four Queens Problem with Domain 1 Restricted)

Ordering	Additions	Pair Checks
Random Order (Mn.)	7	60
Lexicogr./Queue	5	35
Lexicogr./Stack	8	40
Increas. Satisfiab	.2	.29
Decreas. Satisfiab	13	110
Incr. Rel. Satisif.	5	52
Decr. Rel. Satisif	9	65
Increas. Domain J	0	25
Decreas. Domain J	12	87

6 Experiments with Random Problems

6.1 Methods

6.1.1 Problem Generation

Evidence concerning the efficiency of ordering heuristics was obtained experimentally with random CSPs. The main problem sets had six or 12 variables, with a maximum domain size of ten or 12, respectively. The number of binary constraints, the specific variable pairs subject to constraint, the size of each domain, the number of acceptable pairs in each constraint, and the specific value pairs in that constraint were chosen in this order using random methods. This procedure samples from the entire set of possible CSPs within the specified limits, producing problems that are heterogeneous in the features used to order the AC-3 list:

satisfiability, domain size, and degree of constraint graph node. To examine the behavior of heuristics for CSPs with graphs of different density, the factor of constraint number was "blocked" by dividing the range of possible values for each problem size into quarter-ranges. Samples of ten problems with at least one solution and five without solutions were generated for each quarter-range for six-variable problems, and for the lowest two quarter-ranges for 12-variable problems. (This method rarely generates 12-variable problems with solutions from the higher subranges.) CSPs with and without solutions were treated separately because relaxation stops immediately after all values of a domain or constraint have been eliminated. Limits for the number of solutions were set at 200 and 1000 for six- and 12-variable problems, respectively, to balance representativeness and tractibility. (Ten 12-variable problems from the first quarter-range with more than 1000 solutions gave qualitatively similar results, with somewhat smaller differences.)

6.1.2 Evaluation of Performance

Relaxation performance was evaluated using the basic measure of number of value pairs checked for inclusion in a constraint. Other measures were used to identify factors underlying improvement in performance that could be related to the principles described in Section 3: (i) number of list additions, (ii) mean size of domain restriction, (iii) mean location of value deletions in the sequence of constraint checks (the sum of the products of location and number of deletions divided by the total number of deletions). It was also necessary to assess the work required to sort the list initially and to keep it in order after successive relaxations. This is discussed below (Section 7) in connection with the issue of efficient sorting methods for specific heuristics.

For each problem, a baseline for perfomance was obtained by running AC-3 with randomly ordered lists. In these tests, the original list was produced by placing each variable pair in one of the initial set of positions (equal to twice the number of constraints), specified by a pseudorandom number. During relaxation, execution orders were produced by placing each added pair at random in one of q+1 possible positions in a q-element list.

An ordering heuristic partitions the set of variable pairs according to some feature of the problem. Therefore, unless each equivalence class is a singleton, there are many orders consistent with the heuristic at each major step of the procedure, i.e. with the choice of the next variable pair to check. A complete assessment of a heuristic would consider the range of efficiency possible for all such orderings. Since this is not feasible in most cases, it is necessary to rely on random samples, i.e. random orderings of equivalent pairs in the initial list and random insertion into a set of equivalent pairs during execution. For both random lists and lists ordered by heuristics, samples of size ten were used, since preliminary tests showed that these gave sufficiently stable means, compared with samples of 15 or 30.

Differences among orderings were evaluated statistically, using repeated measures analysis of variance (ANOVA) followed by nonorthogonal contrasts for comparisons of individual means. (Here the "mean" refers to a specific problem set, i.e. six- or 12-variable problems; in addition, the number of pair checks for each problem was a mean of ten runs, as just described.) Separate analyses were carried out for six- and 12-variable problems with range of constraints and heuristics as factors. Paired comparison t tests were used to evaluate differences in mean performance (across all constraint ranges) between orderings based on a heuristic and those based on the reversed ordering.

6.2 Results

For all sets of random CSPs, there were marked differences due to ordering the list of variable pairs. For all problem sets, the best orderings reduced the number of pair checks by about 50% in comparison with either reference ordering. (For illustration, Figure 4 gives results for problem sets in the second subrange of constraints.) The main ANOVAs included the two reference orderings: random (RAND) and a lexicographic ordering maintained as a queue (LEX/Q). Three heuristic orderings were also included, based on increasing satisfiability (SAT UP), increasing size of the domain relaxed against (DOM J UP), and decreasing degree of node (DEG DOWN). For both six- and 12-variable problems, the effects of range and heuristic were statistically significant ($p < .001$). Contrasts between the two reference orderings and each heuristic were statistically significant for SAT UP and DOM J UP but not for DEG DOWN. A second ANOVA based on the three satisfiability heuristics, increasing satisfiability, relative satisfiability and increasing value of C_{ij}/d_i, yielded significant effects for range ($p < .05$) and heuristic ($p < .001$). Contrasts between relatively satisfiability and the other two heuristics were statistically significant ($p < .001$); the contrast between the latter was not. The effectiveness of simple satisfiability therefore depends on more than satisfiability *per se*; another factor may be the size of the domains, which tend to be small for low satisfiabilities.

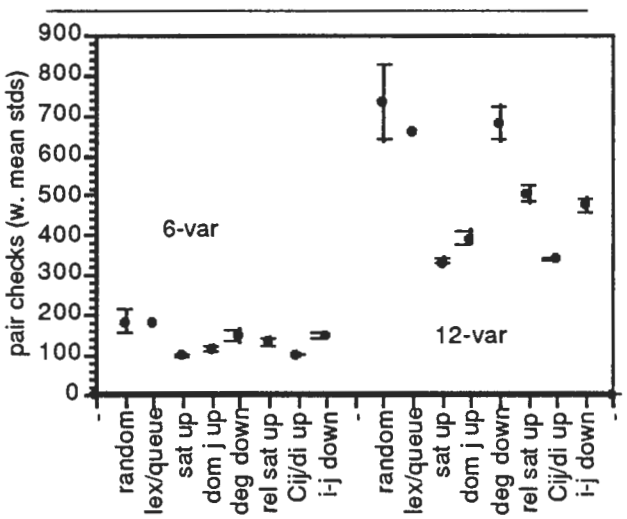


Figure 4. Relaxation effort with different heuristics.

For each heuristic shown in Figure 4, comparisons with the reversed order for six-variable problems yielded

statistically significant differences in favor of the original heuristic ($p < .001$). This shows that each heuristic had some effect on efficiency, even when it could not be distinguished statistically from the reference orderings.

That good heuristics are consistent with the principles of efficient relaxation described in Section 3 is indicated by the analyses of list additions and domain restrictions during relaxation. An ANOVA for list additions based on the same five orderings as the one for pair checks yielded statistically significant effects for both range and heuristic ($p < .001$), and the orthogonal contrasts showed the same pattern of statistically significant results. In fact, the best orderings required only 1/4 and 1/5 of the additions required by the reference orderings for six- and 12-variable problems, respectively. The major part of the improvement in performance was, therefore, due to reduction in list additions. However, small but highly consistent differences ($p < .001$) were found for average size of domain restriction, with largest values associated with the best orderings and smallest values with the reverse orderings. This, together with the finding of more rapid relaxation, reflected in lower mean values for location of deletions ($p < .001$; with a ratio of 2:1 for mean values of the reference orderings versus the best heuristics), suggests that good heuristics also minimized the number of value comparisons. The finding of more rapid relaxation is, of course, a direct demonstration of the ASAP principle.

The above results pertain to problems with solutions. For problems without solutions, the number of pair checks required to remove all elements from a domain or constraint showed the same differences between orderings as for problems with solutions. In the present case, the size of these differences was actually greater: mean performance ratios were 3-4:1 for six-variable problems and 5-10:1 for 12-variable problems. Statistical analysis yielded the same pattern of statistically significant differences as for problems with solutions.

Since Phase 1 of AC-4 uses a list of variable pairs, this part of the algorithm was tested for amenability to ordering heuristics, using the 12-variable problems. The list was ordered lexically or with the DOM J UP heuristic. With the ordering heuristic, fewer pair checks were performed during Phase 1 for both samples of ten problems and more values were deleted ($p < 0.001$ for both effects). Mean pair checks for problems in the second constraint subrange were 1454 for lexical ordering and 1017 for DOM J UP. This far exceeds the numbers required for the entire AC-3 algorithm with these orderings (cf. Figure 4; similar differences between AC-3 and -4 were found for the first subrange) and is due to the need to check each value against all values in adjacent domains, to find the total support for that value. For this reason, no further analysis of AC-4 is presented here, although more extensive examination with respect to average time complexity and effects of ordering would be useful.

7 Effort Required to Keep List Ordered

Although ordering heuristics can have a significant effect on the efficiency of relaxation, costs are incurred in sorting the original set of pairs (*initial ordering*) and in maintaining the proper order as domains are restricted and pairs added during relaxation (*execution ordering*). The latter can affect the overall time complexity of relaxation, because adjustments may be necessary at each step of the procedure. It is therefore worth considering heuristics that order the list initially and then use a placement strategy, such as stacking or queueing, during relaxation, as well as efficient methods for maintaining list order in each phase of the process.

Evidence bearing on the efficiency of initial ordering followed by stacking or queueing was collected for the six- and 12-variable problems of the main experiments. Discussion of these results is limited to SAT UP and DOM J UP, since these were the most promising heuristics overall. For six-variable problems the mean increase in value pair checks for initial ordering alone, compared to initial and execution ordering, was 15% for SAT UP and for DOM J UP. For 12-variable problems the corresponding increases were 50% and 45%. This indicates that execution ordering can have a significant effect on performance, and should be carried out if it can be made efficient.

For initial ordering, $O(e)$ performance can be achieved with a pigeonhole sort. For DOM J UP, this is likely to be sufficient, since the number of pigeonholes equals the maximum domain size, which should not be large. For SAT UP, the number of holes must equal the largest product of two domain sizes, so longer times may be required to scan the list, with useless steps due to empty holes. In the present tests, therefore, the number of pigeonholes was set at 1/5 the maximum. The list was partially sorted in this way, and then each hole was sorted with an insertion sort, which is very fast when no pairs are very out-of-order. For both heuristics $O(e)$ performance was approximated for the two sets of experimental problems.

For DOM J UP, execution ordering is also straightforward and efficient. After each domain restriction, the affected pairs are collected from one pigeonhole in at most $O(e)$ steps and transferred in one further step. Since this is the complexity involved in collecting adjacent pairs and checking that they are not already on the list, the overall complexity of the algorithm can be maintained while keeping the list in order. For SAT UP, the process is complicated by the need to update the representation of constraints, which can take up to $O(ed^2)$ steps. Affected pairs must be rearranged in the list, in addition to adding adjacent pairs, which may require $O(e)O(sort)$ steps. Since relaxation based on DOM J UP appears to be only slightly less efficient than SAT UP in terms of the number of pair checks, the overall efficiency of the former is therefore greater.

That DOM J UP is appreciably faster than LEX/Q overall was confirmed by timing tests on the 12-variable problems. Average CPU time per problem was 3.1 and 4.0 seconds, respectively, and for every problem the difference was in favor of the ordering heuristic.

8 Conclusions

Ordering heuristics can increase the efficiency of relaxation, reducing the number of value pair checks by a factor of two, on average, in comparison with reference orderings. These heuristics order the list of variable pairs so it conforms to

the principles of efficient relaxation discussed in Section 3; the major effect is to minimize list additions. For some heuristics, it is also possible to limit the work involved in establishing and maintaining an effective ordering, so this factor does not outweigh the benefit gained during relaxation. We conclude that ordering by increasing size of the domain relaxed against is an effective general strategy for enhancing performance of algorithms that establish full arc consistency.

Acknowledgements

Richard Lyczak suggested some of the heuristics. Karl Gevecker wrote the program to generate random CSPs.

References

[Cooper, 1989] Cooper, M. C. An optimal k-consistency algorithm. *Artificial Intelligence*, 41:89-95, 1989.

[Dechter and Pearl, 1988] Dechter, R. and J. Pearl. Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence*, 34:1-38, 1988.

[Deville and Van Hentenryck, 1991] Deville, Y. and P. Van Hentenryck. An efficient arc consistency algorithm for a class of CSP problems. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 325-330, Sydney, August 1991. International Joint Committee on Artificial Intelligence.

[Dincbas *et al.*, 1988] Dincbas, M., P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf and F. Berthier. The constraint logic programming language CHIP. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, Tokyo, 1988.

[Freuder, 1978] Freuder, E. C. Synthesizing constraint expressions. *Communications of the ACM*, 21: 958-966, 1978.

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

[Freuder and Wallace, 1991] Freuder, E. C., and R. J. Wallace. Selective relaxation for constraint satisfaction problems. In *Third International Conference on Tools for Artificial Intelligence*, pages 331-339, San Jose, CA, November 1991. IEEE Computer Society Press.

[Gaschnig, 1974] Gaschnig, J. A constraint satisfaction method for inference making. In *Proceedings 12th Annual Allerton Conf. on Circuit System Theory*, pages 866-874, Urbana, IL, 1974.

[Haralick and Elliott, 1980] Haralick, R. M. and G. L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263-313, 1980.

[Mackworth, 1977a] Mackworth, A. K., On reading sketch maps. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 598-606, Cambridge, MA, August 1977. International Joint Committee on Artificial Intelligence.

[Mackworth, 1977b] Mackworth, A. K. Consistency in networks of relations. *Artificial Intelligence*, 8:99-118, 1977.

[Mackworth, 1987] Mackworth, A. K. Constraint satisfaction, In *Encyclopedia of Artificial Intelligence*, S. Shapiro, editor, Vol. 1, pages 205-211. John Wiley & Sons, New York, 1987.

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

[Meiri, 1991] Meiri, I. Combining qualitative and quantitative constraints in temporal reasoning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 260-267, Anaheim, CA, July 1991. American Association for Artificial Intelligence.

[Mohr and Henderson, 1986] Mohr, R. and T. C. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, 28:225-233, 1986.

[Montanari, 1974] Montanari, U. Networks of constraints: Fundamental properties and applications to picture processing. *Information Science*, 7:95-132, 1974.

[Nadel, 1988] Nadel, B. Precision complexity analysis: A framework for deriving exact-case results. Technical Report, Department of Computer Science, Wayne State University, 1988.

[Nadel, 1989] Nadel, B. Constraint satisfaction algorithms. *Computational Intelligence*, 5:188-224, 1989

[Rossi *et al.*, 1989] Rossi, F., V. Dhar and C. Petrie. On the Equivalence of Constraint Satisfaction Problems. Technical Report ACT-AI-222-89, MCC, Austin, TX, 1989.

[Van Hentenryck, 1989] *Constraint Satisfaction in Logic Programming*. MIT Press, Cambridge, MA , 1989.

[Waltz, 1975] Waltz, D. L. Understanding line drawings of scenes with shadows. In *The Psychology of Computer Vision*. P. H. Winston, editor , pages 19-91. McGraw-Hill, New York, NY, 1975.

Extending PATR with Path Patterns and Constraints*

Greg Sidebottom and Fred Popowich
School of Computing Science
Simon Fraser University
Burnaby, British Columbia, V5A 1S6, Canada
Email: {gregory,popowich}@cs.sfu.ca

Abstract

We present an extension of the PATR-II unification based grammar formalism, called PATTERN. PATR-II uses path equations to constrain information contained in feature structures. PATTERN adds a generalization of regular expressions to path equations and path length constraints, effectively incorporating an extension of "functional uncertainty" into PATR-II. We describe in some detail how PATTERN can be implemented in Prolog and compare our implementation with another similar extension to PATR-II. We also give some examples to show that PATTERN can elegantly and declaratively describe some linguistic phenomena.

1 Introduction

Numerous formalisms have been proposed to help linguists formulate their theories for automatic language processing. In recent years, unification based grammar formalisms [Pereira and Warren, 1980; Shieber, 1986] have become popular because they allow linguists to formulate their theories more declaratively than other formalisms, like augmented transition networks (ATNs) [Woods, 1970]. However, the expressive power of these formalisms is restricted in order to make computer processing of theories expressed in them more efficient. As a result, linguistic theories must often be transformed and redundancies must be introduced to be expressed in unification formalisms. Consequently, ad hoc extensions to grammar formalisms are often used in practice to avoid transformations and redundancies. For instance, Prolog predicates are often used in definite clause grammars (DCGs) [Pereira and Warren, 1980] and extensions of DCGs [Johnson and Klein, 1986]. Alternatively, more expressive unification formalisms are needed [Johnson and Rosner, 1989; Balari *et. al*, 1990; Schatz and Lehner, 1990]. Recent work in Constraint Logic Programming (CLP) [Cohen, 1990] promises to efficiently increase the expressive power of unification based grammar formalisms without ad hoc extensions.

This paper describes how to increase the expressive power of the PATR-II (henceforth referred to simply as PATR) formalism [Shieber *et al.*, 1983; Karttunen, 1986; Gazdar and Mellish, 1989]. The main information structures used in the PATR formalism, called feature structures (FSs), are record like structures which are a generalization of terms in standard first order logic. Information in FSs can be accessed by following paths of features, much like components of records are accessed in conventional programming languages. These sequences of feature names leading to information in a FS are called path expressions. FSs can also be unified in a way similar to term unification in logic programming [Lloyd, 1984].

The PATR formalism was developed as a general purpose formalism for implementing other formalisms. In practical terms, PATR is too weak for this since there are other formalisms which often have, as primitive operations, the ability to select an element of a list or access an arbitrarily deeply nested structure. The new formalism described here, called PATTERN, augments PATR with patterns and constraints for path expressions, so it addresses many of the practical weaknesses of PATR in a well founded way. PATTERN path expressions may contain regular expressions, repetition expressions and length constraints, all of which have well defined meanings in formal language theory [Hopcroft and Ullman, 1979]. Moreover, recent work in CLP [Walinsky, 1989] provides some criteria which must be satisfied to ensure that constraints on regular expressions are implemented in a way that coincides with their declarative interpretation.

PATTERN uses a CLP approach to processing constraints on paths by treating them as strings. Constraint processing and CLP have been applied in other logic and unification based approaches to natural language processing. For instance, Tuda *et. al.* [1989] use constraint logic programming to help construct interpretations for sentences containing ambiguous words. Maruyama [1990] uses constraint processing techniques to deal with structural ambiguity. Balari *et al.* [1990] talk about constraint logic grammars. Schatz and Lehner [1990] use Prolog III's [Colmerauer, 1990] Boolean constraint processing capabilities to implement negation and disjunction for finite feature domains in a unification based formalism. Like other methods for expressing sets of FSs disjunctively [Dawar and Vijay-Shanker, 1990; Kasper, 1987; Schatz and Lehner, 1990], PATTERN can be used to elegantly describe alternative structures.

Kaplan and Maxwell [1988] showed the utility of regular path expressions in characterizing linguistic phenomena. However, except for REGDPATR [Carlson, 1988], they have not been incorporated into the latest generation of general purpose unification formalism implementations (such as [Hirsh, 1988; Kilbury, 1990; Yampol and

* This work was supported by the Centre for Systems Science at Simon Fraser University, by the Alberta Research Council, and by NSERC operating grant #OGP0041910.

Karttunen, 1990]). They have been incorporated into REGDPATR by extending FSs so that they allow cyclicity. Our approach does not require a wholesale change to FSs.

The organization of the remainder of the paper is as follows: Section 2 describes how PATTERN augments PATR with path patterns and constraints. Section 3 describes an experimental Prolog implementation of PATTERN. Section 4 gives several example applications of the PATTERN formalism, and section 5 closes with some discussion.

2 PATR with Path Patterns and Constraints

2.1 PATR

The PATR formalism is based on the context free grammar formalism, except instead of atomic non-terminal and terminal symbols, PATR uses FSs. As mentioned earlier, a FS is a kind of generalized record structure which can be unified with other feature structures. Feature structures are often represented in a kind of matrix notation. For instance, consider the following FS:

$$F\left[\begin{array}{l}\text{name:joe}\\ \text{job:}\left[\begin{array}{l}\text{firm:acme}\\ \text{title:supervisor}\end{array}\right]\\ \text{age:A}\end{array}\right].$$

The variable F preceding this FS simply serves to give it a name. Atomic symbols are also considered feature structures. Variables may also occur with no following FS, as the variable A does. These variables name a feature structure which can be anything. The set of *features* defined in a feature structure are the set of atomic symbols which are followed by colons. For instance, the features defined in F are name, job, firm, title, and age. In PATR, feature structures are specified using *path equations*. A *path* is a sequence of features separated by colons. A feature structure is considered a function from paths to other feature structures. For instance, using the FS above, F(name) = joe and F(job:title) = supervisor. Note that F(job) is a non-atomic feature structure, and F(job)(title) is a synonym for F(job:title). In general, path equations can be of the form

[1] $F(f_1{:}f_2{:}\ldots{:}f_n) = G(g_1{:}g_2{:}\ldots{:}g_m)$

where $n, m \geq 0$; F and G are FSs; $f_1, f_2, \ldots, f_n$ and $g_1, g_2, \ldots, g_m$ are features. In this case, the FSs denoted by $F(f_1{:}f_2{:}\ldots{:}f_n)$ and $G(g_1{:}g_2{:}\ldots{:}g_m)$ are unified. Unification succeeds on two FSs F and G if there is no path π such that $F(\pi)$ and $G(\pi)$ are *incompatible*. Two FS are incompatible if they are different atomic FSs or one is atomic and one is not. Unifying FSs may instantiate variables and add paths to them. Note that [1] is an equality, not an assignment. If the two sides were swapped, it would have the same meaning.

FSs are often used to represent lists. For instance, in this paper, a list with elements X, Y, and Z, is represented using the features first and rest, and the atomic FS [], as follows:

[2]
$$\left[\begin{array}{l}\text{first:X}\\ \text{rest:}\left[\begin{array}{l}\text{first:Y}\\ \text{rest:}\left[\begin{array}{l}\text{first:Z}\\ \text{rest:[]}\end{array}\right]\end{array}\right]\end{array}\right].$$

A theory in our version of the PATR formalism is a set of rules of the form

[3] $F_0 \rightarrow F_1, F_2, \ldots, F_n, \{C_1, C_2, \ldots, C_m\}$

where F_i $(0 \leq i \leq n)$ are variables denoting feature structures and C_j $(1 \leq j \leq m)$ are path equations. The order in which the path equations appear does not matter, but to use the rule, they all must be true. Rules can be used to describe both the lexical information associated with words, and the syntactic and semantic structure of phrases. For instance, the rule:

Word → {Word(lex)=he, Word(num)=sing, Word(cat)=pro}

defines a lexical entry for the word 'he'. The rule:

S → NP, VP, {S(cat)=s, NP(cat)=np, VP(cat)=vp,
S(sub)=NP(ref), NP(num)=VP(num)}

defines some of the syntactic and semantic relationships between the constituents of simple sentences composed of a noun phrase followed by a verb phrase. Specifically, the subject of the sentence (S) is determined from the referent of the noun phrase (NP), and the NP and verb phrase (VP) exhibit number agreement.

2.2 The Extension

PATTERN generalizes the paths used in path equations and allows constraints other than path equations in rules. Specifically, the following two extensions are made:

1. any of the features f_i or g_j in [1] above can be replaced by a pattern expression, and
2. any C_j in [3] can be a path equation, set membership, or length constraint.

The remainder of this section defines pattern expressions, set membership constraints, and length constraints.

Pattern expressions are a generalization of regular expressions. As usual, regular expressions are used to denote regular sets; ε is the empty string; regular expressions are constructed from a set Σ of symbols using the alternation (+), concatenation ($\cdot$), and Kleene closure (*) operators; $|\sigma|$ is the length of $\sigma \in \Sigma^*$; and $L(e)$ is the set of strings denoted by an expression e. We also use the notation e^+ where $L(e^+) = L(e \cdot e^*)$ and e^n (non-negative integer n) where $L(e^0) = \{\varepsilon\}$ and $L(e^{i+1}) = L(e \cdot e^i)$ for $i > 0$.

The constraints in PATTERN are based on those used in CLP(Σ^*) [Walinsky, 1989] which provides membership constraints on regular sets over a finite set Σ. An important feature of CLP(Σ^*) is that logic variables can be used to represent a named element of Σ^* in expressions. For instance, $L(S \cdot S) = \{\sigma \cdot \sigma \mid \sigma \in \Sigma^*\}$. The power of this feature becomes apparent when variables are used in more than one constraint. For instance, the pair of constraints

[4] $\{X \in A \cdot x \cdot W1, Y \in W2 \cdot x \cdot A\}$

force X and Y to denote strings containing a distinguished 'x', where X's prefix before the 'x' is the same as Y's suffix after the 'x'.

Like Kaplan and Maxwell [1988], we interpret path equations of the form $F(\pi{:}e{:}\rho) = G(\tau)$ where π, ρ, and τ are paths and e is a pattern expression, as the disjunction of

$F(\pi:\sigma:\rho) = G(\tau)$[1] for each $\sigma \in L(e)$. The possibility of using variables as in [4] above gives PATTERN more power than Kaplan and Maxwell's formalism or REGDPATR [Carlson, 1988].

PATTERN extends CLP(Σ^*) constraints in four ways. First, since PATTERN uses components of strings in path expressions, the set Σ over which strings range is the infinite set of all possible feature names. However, the complete set of features used in any particular PATTERN description is finite, so we will assume later that Σ is a finite set. Second, PATTERN has the complement operator '-', where $L(e_1 - e_2) = L(e_1) \setminus L(e_2)$ and $L(-e) = \Sigma^* \setminus L(e)$. Third, PATTERN has repetition expressions of the form e^f, where f is a non-negative integer. Fourth, PATTERN has relational constraints ($=, \neq, <, \leq$) on integer expressions which may involve string length expressions.

The list constraints in Prolog III [Colmerauer, 1990] provide some functionality similar to the expression constraints in PATTERN. Prolog III allows list concatenation constraints and list length constraints. The main difference is that Prolog III uses equality and disequality constraints on list expressions directly (eg. X·Y = Z), instead of set membership constraints.

Path patterns make it possible to represent disjunctive constraints on FSs. For instance, if X is a FS representing a list using the same scheme as in [2] above, the path equation {X(rest*:first) = Y} selects an element out of the list and unifies it with Y. A more complicated example is {$1 \leq |S| \leq 4$, S $\in$ rest*, X(S:first) = Y}, which unifies Y with the one of the second, third, fourth or fifth elements of X. When path patterns and constraints are used in conjunction with rules, fairly complicated data structure accesses and transformations can be succinctly described.

```
1.  W ∈ E ← in(E, W, []).
2.  in(E) → {var(E), !}, prefix(E).
3.  in(X) → {inΣ(X)}, [X].
4.  in([]) → [].
5.  in([X|L]) → [X], in(L).
6.  in(E1·E2) → in(E1), in(E2).
7.  in(E1+E2) → in(E1) | in(E2).
8.  in(E*) → in([]) | in(E·E*).
9.  in(E+) → in(E·E*).
10. in(E0) → [].
11. in(EN) → {N > 0, M is N - 1}, in(E), in(EM).
12. prefix([]) → [].
13. prefix([X|W]) → [X], prefix(W).
```

Figure 1 Prolog code for PATTERN's membership constraints

[1] We use paths and strings of symbols interchangably.

3. An Experimental Implementation in Prolog

The Prolog source code for PATTERN's set membership constraints is given in figure 1[2]. The complement operator has not been implemented. However, negation of set membership constraints can be used in many instances where the complement operator is needed.

Strings are represented by lists of terms, with [] representing the empty string ε. For clarity, we have used the common mathematical notation as much as possible. To actually execute this program in a Prolog system, minor modifications would have to be made to remove special symbols and superscripts.

The first line of the program is a clause which defines the set membership predicate ($\in$/2) in terms of the definite clause grammar (DCG) given in the remaining lines. Most Prolog systems automatically convert DCGs to Prolog; see [Sterling and Shapiro, 1986] for more details.

The DCG defines how to parse a string according to an expression. The production on line 2 takes care of the case where the expression E is a variable. It binds E to some prefix of the string being parsed using the predicate prefix/3, defined by the productions given in lines 12 and 13. The cut is used in line 2 because variables are intended to have only the meaning defined by line 2, but would match all the other productions for in/3. The production on line 3 takes care of the case where the expression X is a constant in the set Σ of terminal symbols. The predicate call inΣ(X) simply tests that X is some non-variable term which is not a list and whose functor is not any of the ones used to construct expressions (eg. not ·/2, +/2, */1 or +/1). Lines 4 and 5 cover the case where the expression is a string (represented by a list, as mentioned earlier) and ensure that the input string matches the expression. Lines 6 and 7 take care of the concatenation and alternation operators in the obvious way. Line 8 defines the Kleene closure operator in the usual recursive fashion and line 9 defines +/1 in the standard way. Finally, lines 10 and 11 define repetition patterns recursively.

String length constraints are implemented in a straight forward way by generating lists of variables with appropriate lengths. Then, these lists are matched with the strings to be constrained.

Clearly, the power of PATTERN constraints is far greater than that of regular constraints such as those in CLP(Σ^*). Given an order on Σ, the set of solutions for the variables in a set of constraints can be enumerated. However, this enumeration process is not guaranteed to terminate since many constraints have an infinite number of solutions.

Walinsky [1989] proves for regular expression constraints over a finite set Σ that constraints of the form $\sigma \in e$ have a finite number of solutions when either σ is a ground string or e contains no variables or closure operators. Still assuming Σ is finite, this result can be generalized to where $|\sigma|$ has a finite upper bound or e contains no variables,

[2] We are indebted to Cliff Walinsky for making available a Prolog regular set membership constraint system on which this code is based.

closure operators, or complement operators. For constraints with a finite set of solutions, it is safe to implement the complement operator using negation as failure [Lloyd, 1984].

One way to implement PATTERN constraints in a logic programming language like Prolog is to find a computation rule which delays evaluating constraints when they have an infinite number of solutions. As the previous discussion shows, this is possible for the kinds of constraints used in PATTERN system, though some programs may flounder in a manner similar to a safe implementation of negation as failure [Naish, 1985].

It is often possible avoid the floundering problem by setting bounds on the length of the strings generated by pattern expressions. In many cases, bound setting can be automated by analyzing PATTERN rules and the input sentence. For instance, most PATTERN grammars which use lists represented by the scheme in [2] above will often only generate lists whose lengths are bounded by the size of the input sentence. These bounds can be used to constrain the length of strings generated by patterns such as rest*. We conjecture that this kind of analysis is sufficient for the effective use of most PATTERN grammars.

PATTERN is implemented by a simple modification to the left corner parser from [Gazdar and Mellish, 1989]. All that is necessary is to modify the third clause of the denotes/2 predicate to expand regular expressions embedded in paths using the in/3 predicate defined in figure 1. We have not implemented the specialized computation rule or automatic bound setting. For now, we use a global bound to limit the length of strings generated by all patterns.

The computational complexity of analyzing sentences with PATTERN grammars is very high. Since PATR is Turing complete, the problem is undecidable in general. The problem of unifying disjunctive feature structures, which is subsumed by the problem of solving PATTERN constraints, is *NP*-complete [Kasper and Rounds, 1986]. However, in our experience, the time to analyze sentences using PATTERN grammars is similar to that of using plain PATR grammars.

4. Applications

4.1 Anaphoric Dependency in Discourse

Johnson and Klein [1986] show how a DCG formalism extended with FSs can be used to implement the theory of inter- and intra-sentence anaphoric dependency from Discourse Representation Theory. Figure 2 compares the PATTERN and DCG rules for the lexical entry for the word 'she'. Johnson and Klein use both Prolog lists and FSs in their implementation, as well as calls to the Prolog predicate member/2. The calls to member/2 are used to select arbitrary appropriate referents for pronouns in a list of enclosing spaces. In PATTERN, we formulate lists using the scheme of [3] and use constraints involving the pattern rest* to achieve the same effect as the calls to member. It should be noted that the third and fourth constraints in the PATTERN rule can be combined into one constraint:

W:syn:index = W:sem:in:rest*:first:rest*:first.

Two constraints are used to correspond exactly with Johnson and Klein's rule, where two calls to member/2 are required.

The remainder of the grammar given by Johnson and Klein is translated to PATTERN in a straight forward way, though some transformations are awkward. However, where PATTERN is awkward, Johnson and Klein use Prolog. We feel it is to PATTERN's credit that Discourse Representation Theory can be expressed within it. Both grammars can be used to analyze discourses such as "A woman chased every donkey. Every boy saw her." to discover that 'her' refers to 'a woman'. They can also be used to discover that "A woman chased every donkey. Every boy saw it." is meaningless because 'it' has no referent.

4.2 Free Word Order

In [Carlson, 1988], a REGDPATR description, called RUG, is given which can be used to analyze free word order languages. The rules in RUG set up a doubly linked list whose elements represent the words in the sentence. The list is constructed using the features 'left' and 'right' to indicate, respectively, the word to the left and right of the current word. Of particular interest in RUG is the use of regular path expressions to let verbs select their complements from an arbitrary length list. The list is stored an arbitrary number of list elements to the left or right of the list element for the verb. The key patterns used in paths are named AnyVComp, GrmFn and Domain. PATTERN can be used to elegantly implement RUG. AnyVComp, which is used to select an element out of a list, is the same as 'vcomp*' in PATTERN. GrmFn (for grammatical function) is the same as 'comp1 + comp2'. Domain, which is used to look one or more list elements to the left or right of the verb, is the same as 'left+ + right+'.

```
W → {  W:lex = she,
       W:cat = np,
       W:sem:in:rest*:first = Space,
       W:syn:index = Space:rest*:first,
       W:syn:index:type = fem,
       W:sem:scope:in = W:sem:in,
       W:sem:out = W:sem:scope:out}.
```

(a) PATTERN

```
w(W) → {  W:lex = she,
          W:cat = np,
          member(Space, W:sem:in),
          member(NP:syn:index, Space),
          W:syn:index:type = fem,
          W:sem:scope:in = W:sem:in,
          W:sem:out = W:sem:scope:out}.
```

(b) DCG with FSs

Figure 2 Comparison of PATTERN with DCG extended with FS

4.3 Functional Uncertainty

Kaplan and Maxwell [1988] give examples in Lexical Functional Grammar [Kaplan and Bresnan, 1982] which show how regular feature sets (under the name of "functional uncertainty") can be used to elegantly describe long distance dependencies. Here, we put these ideas into the more general PATTERN framework. A difficult problem in handling topicalized sentences is that there can be an unbounded number of nested clauses. For instance, Kaplan and Maxwell [1988] use examples of the form (Mary, John claimed that Bill said that ... that Henry telephoned yesterday). In PATTERN, the solution to this problem uses constraints of the form: {F(topic) = F(comp*:GF), GF ∈ subj + obj + obj2 + xcomp}. The expression 'comp*' accesses an arbitrary embedded clause and the constraint 'GF ∈ subj + obj + obj2 + xcomp' forces the variable GF to denote one of the primitive grammatical functions. We note in this context that our current PATTERN implementation's global bound on string lengths seems physiologically plausible, since people can only keep track of a bounded number of dependencies.

5. Discussion

5.1 Comparison with REGDPATR

There are two fundamental differences between PATTERN and REGDPATR. First, PATTERN as with most FS formalisms, uses only atomic symbols for features while REGDPATR allows FSs to act as features. We will call these FSs which act as features *feature structured features* (FSFs). In REGDPATR, FSFs are interpreted as disjunctions of paths. Second, REGDPATR requires cyclic feature structures to implement the Kleene closure and iteration operators. For instance, the expression 'a*' in PATTERN is equivalent to the cyclic FSF

$$X\begin{bmatrix}\text{final:t}\\ \text{a:X}\end{bmatrix}$$

in REGDPATR. When encountering this FSF during unification, REGDPATR must transform it into a path. To do this, REGDPATR either takes the 'final' or the 'a' feature. Taking the former terminates the path and taking the latter adds an 'a' to the path and loops.

From the description given in [Carlson, 1988], it appears that PATTERN can directly represent the pattern expressions available in REGDPATR. REGDPATR has patterns of the form (anyof A B), which are expressed by 'A + B' in PATTERN. REGDPATR patterns of the form (noneof A B), which require A and B to be atomic FSs, are generated by S in {not S ∈ A+B}. REGDPATR also has something described as "iteration of paths," which apparently means patterns of the form 'E^N' or something like those generated by S in {2≤|S|≤4, S in a*} in PATTERN.

In conjunction with unification, REGDPATR uses "unit path unification" to search for alternative FSs which contain no FSFs. It should be noted that the unification/unit path resolution algorithm is incomplete in the sense that it may fail to detect that two feature structures are not unifiable. This is similar to the fact that our implementation of PATTERN, when augmented with a termination safe computation rule, may terminate with floundered constraints which may be inconsistent.

5.2 Conclusion

The PATTERN formalism can be used to elegantly describe a variety of linguistic phenomena. Its pattern expressions and string constraints make it especially suited for describing disjunctive sets of FSs and access and transformation operations on data structures. PATTERN's basis in formal language theory gives its constructs a well defined declarative semantics and a concise understandable mathematical notation. Though the formalism is so expressive that its implementation may have termination and efficiency problems, we argued that it can be implemented to perform reasonably well for practical purposes. Our initial experience with an experimental implementation of PATTERN supports this conjecture. Thus, we conclude that the PATTERN formalism is worthy of further study.

References

[Balari *et. al*, 1990] Balari, S., L. Damas and N. Moreira (1990) "CLG(n): Constraint Logic Grammars." In Proc. The 13th International Conference on Computational Linguistics, Helsinki, Finland, vol. 3, pp. 7-12.

[Carlson, 1988] Carlson, L. (1988) "RUG: Regular Unification Grammar." In Proc. The 12th International Conference on Computational Linguistics, Budapest, pp. 102-105.

[Cohen, 1990] Cohen, J. (1990) "Constraint Logic Programming," *Communications of the ACM*, 33 (7), pp. 52-68.

[Colmerauer, 1990] Colmerauer, A. (1990) "An Introduction to Prolog III," *Communications of the ACM*, 33 (7), pp. 69-90.

[Dawar and Vijay-Shanker, 1990] Dawar, A. and K. Vijay-Shanker (1990) "An Interpretation of Negation in Feature Structure Descriptions," *Computational Linguistics*, 16 (1), pp. 11-20.

[Gazdar and Mellish, 1989] Gazdar, G. and C. Mellish (1989) *Natural Language Processing in Prolog*, Addison-Wesley, Don Mills, ON.

[Hirsh, 1988] Hirsh, S. (1988) "P-PATR: A Compiler for Unification-based Grammars." In *Natural Language Understanding and Logic Programming II*, V. Dahl and P. Saint-Dizier (ed.), Elsevier Science Publishers B.V., North-Holland.

[Hopcroft and Ullman, 1979] Hopcroft, J. E. and J. D. Ullman (1979) *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Don Mills, ON.

[Johnson and Klein, 1986] Johnson, M. and E. Klein (1986) "Discourse, Anaphora and Parsing." In Proc. *The*

11th International Conference on Computational Linguistics, Bonn, pp. 669-675.

[Johnson and Rosner, 1989] Johnson, R. and M. Rosner (1989) "A Rich Environment for Experimentation with Unification Grammars." In Proc. *ACL Fourth European Conference*, Manchester, pp. 182-189.

[Kaplan and Bresnan, 1982] Kaplan, R. M. and J. Bresnan (1982) "Lexical-Functional Grammar: A Formal System for Grammatical Representation." In *The Mental Representation of Grammatical Relations*, J. Bresnan (ed.), MIT Press, Cambridge, pp. 173-281.

[Kaplan and Maxwell, 1988] Kaplan, R. and J. Maxwell (1988) "An Algorithm for Functional Uncertainty." In Proc. *The 12th International Conference on Computational Linguistics*, Budapest, Hungary, pp. 297-302.

[Karttunen, 1986] Karttunen, L. (1986) "D-PATR: a Development Environment for Unification-Based Grammars." In Proc. *The 11th International Conference on Computational Linguistics*, Bonn, pp. 74-80.

[Kasper and Rounds, 1986] Kasper, R. and W. Rounds (1986) "A Logical Sematics for Feature Structures." In Proc. *The 24th Annual Meeting of the Association for Computational Linguistics*, pp. 257-266.

[Kasper, 1987] Kasper, R. T. (1987) "A Unification Method for Disjunctive Feature Descriptions." In Proc. *25th Annual Meeting of the ACL*, Stanford, pp. 235-242.

[Kilbury, 1990] Kilbury, J. (1990) "QPATR and Constraint Threading." In Proc. *The 13th International Conference on Computational Linguistics*, H. Karlgren (ed.), Helsinki, vol. 3, pp. 382-384.

[Lloyd, 1984] Lloyd, J. W. (1984) *Foundations of Logic Programming*, Symbolic Computation, D. W. Loveland (ed.), Springer-Verlag, New York..

[Maruyama, 1990] Maruyama, H. (1990) "Structural Disambiguation with Constraint Propagation." In Proc. *28th Annual Meeting of the ACL*, Pittsburgh, pp. 31-38.

[Naish, 1985] Naish, L. (1985) "Negation and Control in Prolog." In *Lecture Notes in Computer Science 238*, G. Goos and J. Hartmanis (ed.), Springer-Verlag, New York.

[Pereira and Warren, 1980] Pereira, F. C. N. and D. Warren (1980) "Definite Clause Grammars for Language Analysis—a Survey of the Formalism and a Comparision with Augmented Transition Networks," *Artificial Intelligence*, **13**, pp. 231-278.

[Schatz and Lehner, 1990] Schatz, U. and C. Lehner (1990) "Implementation of the Comprehensive Unification Formalism." Report #CIS-Bericht-90-30, Centrum für Informations- und Sprachverarbeitung, Universität München.

[Shieber *et al.*, 1983] Shieber, S., *et al.* (1983) "The Formalism and Implementation of PATR-II." In *Research on Interactive Acquisition and Use of Knowledge*, B. Grosz and M. Stickel (ed.), SRI International, Menlo Park, CA, pp. 39-79.

[Shieber, 1986] Shieber, S. (1986) *An Introduction to Unification-Based Approaches to Grammar*, CSLI Lecture Notes #4, Stanford CA.

[Sterling and Shapiro, 1986] Sterling, L. and E. Shapiro (1986) *The Art of Prolog: Advance Programming Techniques*, MIT Press, Cambridge, MA.

[Tuda *et al.*, 1989] Tuda, H., K. Hasida and H. Sirai (1989) "JPSG Parser on Constraint Logic Programming." In Proc. *ACL Fourth European Conference*, Manchester, pp. 95-102.

[Walinsky, 1989] Walinsky, C. (1989) "Constraint Logic Programming with Regular Sets." In Proc. *Sixth International Conference on Logic Programming*, Lisbon, Portugal, pp. 181-196.

[Woods, 1970] Woods, W. A. (1970) "Transition Network Grammars for Natural Language Analysis," *Communications of the ACM*, **13**, pp. 591-606.

[Yampol and Karttunen, 1990] Yampol, T. and L. Karttunen (1990) "An Efficient Implementation of PATR for Categorial Unification Grammar." In Proc. *The 13th International Conference on Computational Linguistics*, Helsinki, Finland, vol. 2, pp. 419-424.

Structure Identification in relational data *

Rina Dechter
Information and Computer Science
University of California
Irvine, CA 92717
dechter@ics.uci.edu

Judea Pearl
Cognitive Systems Laboratory
Computer Science Department
University of California
Los Angeles, CA 90024
judea@cs.ucla.edu

Abstract

This paper presents several investigations into the prospects of identifying meaningful structures in empirical data, structures that permit effective organization of the data to meet requirements of future queries. We propose a general framework whereby the notion of identifiability is given a precise formal definition, similar to that of learnability. Using this framework, we then address the problem of expressing a given relation as a k-Horn theory and, if this is impossible, finding a best k-Horn approximation to the given relation.

1 Introduction

Discovering meaningful structures in empirical data has long been regarded as the hallmark of scientific activity. Yet, despite the mystical aura surrounding such discoveries we often find that computational considerations of efficiency and economy play a major role in determining what structures are considered meaningful by scientists. Along this vein, we address the task of finding a computationally attractive description of the data, a description that, both, is economical in storage, and permits future queries to be answered in a tractable way.

Invariably, the existence of such a desirable description rests on whether the dependencies among the data items are decomposable into local, more basic dependencies, possessing some desirable features. A classical example would be to find a finite state machine (with the least number of states) that accounts for observed dependencies among successive symbols in a very long string. In more elaborate settings the dependencies can form a graph (as in the analysis of Markov fields) or a hypergraph (as in relational databases), and the task is to find the topology of these structures. Structure identification includes such tasks as finding effective representations for probability distributions, finding economical decompositions of database schema, finding simple Boolean expressions for truth tables, or finding logical theories that render subsequent processing tractable.

*This work was supported in part by the Air Force Office of Scientific Research, AFOSR 900136 and by NSF grant IRI-9157936.

Despite the generality of the task at hand, very few formal results have been established, and these were primarily confined to probabilistic analysis [Chow and Liu, 1968; Pearl and Verma, 1991]. In this paper we focus on categorical data and categorical descriptions of the data. Given a relation ρ in the form of an explicit listing of the tuples of ρ, we ask whether we can find a more desirable description of ρ, say a constraint network possessing desirable topological features, or a logical theory possessing desirable syntactic features (e.g., Horn theories). The former is treated in a recent report [Dechter and Pearl, 1991] and the latter in section 3. In both cases the desirable features would be such that facilitate efficient query processing routines.

We view this task as an exercise in automatic **identification**, because our main concern will be to recognize cases for which desirable descriptions exist and to identify the parameters of at least one such description. Thus, we explore the existence of a tractable identification procedure that takes data as input, returns a theory and works in time polynomial in the size of the input. Given that the data was generated from a theory that has a desirable structure, our procedure should identify the underlying structure if it is unique, or an equivalent structure in case it is not unique. Conversely, if the data does not lend itself to effective organization, we wish our procedure to acknowledge this fact, so as to save further explorations. An additional requirement is sometimes imposed on the procedure, to identify a "best" approximated theory, in case an exact desirable theory does not exist. We call this latter requirement "strong identifiability".

Our analysis bears close relationships to that of Selman and Kautz [Selman and Kautz, 1991], where theory formation is treated as a task of "knowledge compilation". The main difference between the two approaches is that Selman and Kautz begin with a preformed *theory* in the form of a (reasonably sized) set of clauses, while we start with the bare observations, namely, a (reasonably sized) set of tuples which represent the models of the desired theory. This enables us to easily project the relation onto subsets of variables and solve subtasks which would be intractable had we started with a clausal theory. Another difference is that we require definite determination of whether the theory approximates or describes the data.

This paper is organized as follows: Section 2 introduces a general framework of the identification task. We define weak and strong notions of identifiability and compare them to Valiant's [Valiant, 1984] notion of learnability using familiar examples. Section 3 focuses on identifying Horn theories and shows that k-Horn theories (in which every clause contain at most k literals) can be identified and updated in polynomial time, when k is bounded. All theorems will be stated without proofs, which can be found in [Dechter and Pearl, 1991].

2 Preliminaries and Basic Definitions

2.1 Theories: Networks and Formulas

We denote propositional symbols, also called *variables*, by upper case letters $P, Q, R, X, Y, Z, ...$, propositional literals (i.e. $P, \neg P$) by lower case letters $p, q, r, x, y, z, ...$ and disjunctions of literals, or *clauses*, by $\alpha, \beta....$ The complement operator $\sim$ over literals is defined as usual: If $p = \neg Q$, then $\sim p = Q$, If $p = Q$ then $\sim p = \neg Q$. A *formula* in conjunctive normal form (CNF) is a set of clauses, $\varphi = \{\alpha_1, ..., \alpha_t\}$ and it denotes their conjunction. The *models* of a formula φ, $M(\varphi)$, is the set of all satisfying truth assignments to all its symbols. A clause α is *entailed* by φ, written $\varphi \models \alpha$, iff α is true in all models of φ. A clause α is a *prime implicant* of φ iff $\varphi \models \alpha$ and $\nexists \beta \subseteq \alpha$ s.t. $\varphi \models \beta$. A Horn formula is a CNF formula whose clauses all have at most one positive literal. A k-CNF formula is one in which clauses are all of length k or less, and a k-Horn formula is defined accordingly.

Given a clause α we denote by $base(\alpha)$ the set of all propositional symbols on which α is defined. For instance, if $\alpha = \{P \vee \neg Q \vee R\}$ then $base(\alpha) = \{P, Q, R\}$. To characterize the structure of a formula φ we define its *scheme* to be the set of variables on which clauses are defined. Formally:

Definition 1 *(Scheme)*
Let $\varphi = \varphi(x_1, ..., x_n) = \{\alpha_1, ..., \alpha_r\}$, *then*

$$scheme(\varphi) = \{base(\alpha_j) | 1 \leq j \leq r\}. \quad (1)$$

Example 1 *Consider the formula*

$$\varphi = \{(\neg P \vee Q \vee R), (P \vee S), (\neg P \vee \neg S), (\neg P \vee R)\}. \quad (2)$$

In this case,

$$scheme(\varphi) = \{\{P, Q, R\}, \{P, S\}, \{P, R\}\}, \quad (3)$$

We next define the notions of *constraint networks* and *relations* which parallel the notions of formulas and their satisfying models for the case of multi-valued variables. A *relation* associates multi-valued variables with a set of tuples specifying their allowed combinations of values. A *constraint network* is a set of such relations, each defined on a subset of the variables and, together, are taken as conjunction of constraints, namely, they restrict value assignments to comply with each and every constituent relation. The theory of relations was studied extensively in the database literature [Maier, 1983].

Definition 2 *(Relations and Networks)*
Given a set of multi-valued variables $X = \{X_1, ..., X_n\}$, *each associated with a domain of discrete values,* $D_1, ..., D_n$, *respectively, a* **relation** *or a* **constraint** $\rho = \rho(X_1, ..., X_n)$ *is a subset of value assignments to the variables in* X,

$$\rho = \{(x_1, ..., x_n) | (x_1, ..., x_n) \in \{D_1 \times D_2 \times ... \times D_n\}\}. \quad (4)$$

A **constraint network** *over* X, $N(X)$, *consists of a set of such relations* $\rho_1, ..., \rho_t$ *each defined on a subset of variables* $S_1, .., S_t, S_i \subseteq X$. *The set* $S = \{S_1, .., S_t\}$ *is called the* **scheme** *of the constraint network, denoted* $scheme(N)$. *The network* N *represents a unique relation* $rel(N)$ *defined on* X, *which stands for all consistent assignments, namely:*

$$rel(N) = \{\overline{x} = (x_1, ..., x_n) | \forall S_i \in S, \Pi_{S_i}(\overline{x}) \in \rho_i\}. \quad (5)$$

where $\Pi_{S_i}(\overline{x})$ *denotes the projection of* $\overline{x}$ *onto* $S_i \subseteq X$. *If* $rel(N) = \rho$ *we say that* N **describes** ρ.

Clearly, any CNF formula can be viewed as a special kind of constraint network, where the domains are bi-valued, and where the models of each clause specify a constraint on the variables contained in that clause. Accordingly, we say that a bi-valued relation $\rho = \rho(X_1, ..., X_n)$ is **described** (or **represented**) by a formula $\varphi = \varphi(x_1, ..., x_n)$ iff $M(\varphi) = \rho$. We will use the term *theory* to denote either a network or a formula and, correspondingly, use $scheme(T)$ and $M(T)$ (or $rel(T)$).

When considering ways of approximating a relation ρ by a theory T we will examine primarily upper bound approximations, namely, theories T such that $\rho \subseteq M(T)$.

Definition 3 *A theory* $T \in C$ *is said to be a* **tightest** *approximation of* ρ *relative to a class* C *of theories if* $\rho \subseteq M(T)$, *and there is no* $T' \in C$ *such that* $\rho \subseteq M(T') \subset M(T)$.

Example 2 *The following relation:*

P	Q	R	S
1	0	1	0
1	1	1	0
0	1	0	1
0	0	1	1
0	1	1	1
0	0	0	1

can be defined by the network:

P	Q	R
1	0	1
1	1	1
0	1	0
0	0	1
0	1	1
0	0	0

P	S
0	1
1	0

P	R
0	0
0	1
1	1

Being bi-valued, this relation can also be described by the formula:

$$\varphi = \{(\neg P \vee Q \vee R), (P \vee S), (\neg P \vee \neg S), (\neg P \vee R)\}. \quad (6)$$

2.2 Identifiability

We are now ready to give a formal definition of identifiability – a property intrinsic to any class of theories and which governs our ability to decide whether a given

relation ρ has a description within the class or not. As a preliminary and trivial example, we will then show (in subsection 2.4) that the class of k-CNF formulas is identifiable only for $k = 2$, namely, there is no tractable way of deciding whether an arbitrary relation ρ has a description as a k-CNF formula, unless $k = 2$. The class of 2-CNF theories, however, will turn out to be strongly identifiable, namely, not only can we decide the existence of a 2-CNF description, but we can also produce such a description if it exists, or, produce the tightest 2-CNF theory if a precise description does not exists (hence the term "strong".)

To motivate the definition below we should notice that the decisions above depend on what we know apriori about the observed relation ρ. For example, were we given assurance that ρ has a description in k-CNF, it would be easy to produce one such a description. Thus, it is necessary to define the notion of identifiability relative to a background class C' of theories from which ρ is chosen. We will adopt the convention that unless stated otherwise, C' is presumed to be the class of all theories, namely, ρ is arbitrary.

Definition 4 *: (Identifiability)*
A class of theories C is said to be **identifiable** *relative to a background class C', iff:*

1. *(Recognition) For every relation ρ that is describable by some theory T in C', there is an algorithm A, polynomial in $|\rho|$, that determines if ρ has a description in C, and*
2. *(Description) If the answer to (1) is positive, A finds one theory of $T \in C$ that describes ρ, (i,e., $\rho = M(T)$)*
   C is said to be **strongly** *identifiable if, in addition to (1) and (2) above:*
3. *(Tightness) A always finds a theory T_0 in C that is a tightest approximation of ρ.*

By convention, a class in which the recognition or description tasks are NP-hard will be defined as non-identifiable. Note however that the complexity of A is measured relative to the size of ρ, and not relative to the size of its description T. Thus, the notion of identifiability will be applicable to highly constrained theories where the number of distinct observations grows polynomially with the number of variables.

2.3 An example: Identifiability of k-CNF theories.

Let C' be the set of all theories defined on n binary variables. Consider whether the class $C_k \subset C'$ of relations expressible by k-CNF theories is identifiable relative to C'. Although we have algorithms for meeting requirement (3) (and hence (2)) of constructing a tightest k-CNF approximation for any given relation ρ, we do not have an effective way of testing whether this approximation represents the relation ρ exactly, or a superset thereof. Even generating a single model of a tightest k-CNF theory is an NP-hard problem for $k > 2$. We thus conclude that C_k is not identifiable[1].

[1]The non-existence of a tractable procedure for testing exact match with ρ is based on an unpublished theorem conveyed to us by J. Ullman.

Now consider the case where the background class C' is known in advance to consist of k-CNF theories, namely, ρ has a k-CNF description. It is easy then to identify one k-CNF theory which describes ρ. We simply project ρ onto every subset of k or less variables and, for every r-tuple t, $(r \leq k)$ that is not found in the projection of ρ on $X_1, X_2, ..., X_r$ we introduce the clause $(x_1 \vee x_2 \vee ...x_r)$, with x_i being a positive literal iff x_i is false in t. This can be accomplished in time proportional to $|\rho|(2n)^k$.

The theory found can be shown (see section 3) to be a tightest approximation of ρ relative to C_k and, since ρ is assured to have a description in C_k, this theory must be a precise description of ρ. We thus conclude that C_k is strongly identifiable relative to itself.

As a final variant of this example, consider the class of 2-CNF theories. This class is identifiable relative to any arbitrary C', and the reason is as follows: Given an arbitrary relation ρ, we can find a tightest 2-CNF approximation τ of ρ by the projection method described above. There remains to determine whether τ represents ρ precisely. This last task can be accomplished by simply comparing the size of $M(\tau)$ to that of ρ. If the two sizes are equal, τ is obviously a description of ρ, because $M(\tau)$ contains ρ. The distinct feature that renders 2-CNF theories identifiable (unlike k-CNF, $k > 2$) is the tractability of the size-comparison task. A recent result [Dechter and Itai, 1991] states that for every theory T satisfiable in time t, deciding whether $|M(T)| > c$ takes time $O(ct)$. Now , since 2-SAT is satisfiable in polynomial time, testing $M(\tau) > |\rho|$ can also be accomplished in polynomial time.

2.4 Identifiability vs. Learnability

There is a strong resemblance between the notion of identifiability and that of learnability [Valiant, 1984]. If we associate theories with concepts (or functions) and the models of a theory with the learning examples, we see that in both cases we seek a polynomial algorithm that will take in a polynomial number of examples and will produce a concept (or a function) consistent with those examples, from some family of concepts C. Moreover, it is known that in order for a family C to be learnable (with one-sided errors) it must be closed under intersection, and the algorithm must produce the tightest concept in C consistent with the observations [Natarajan, 1987]. This is identical to condition (3) of strong identifiability.

The main difference between the problems described in this paper and those addressed by Valiant's model of learning is that in the latter we are given the concept class C and our task is to identify an individual member of C that is (probably) responsible for the observed instances (in the sense of assuming a small probability of error on the next instance). By contrast, in structure identification we are not given the concept class C. Rather, our objective is to decide whether a fully observed concept ρ, taken from some broad class C' (e.g., all relations) is also a member of a narrower class C

of concepts, one that possesses desirable syntactical features (e.g., 2-CNF, a constraint-tree, or a Horn theory). Thus, the task is not to infer the semantic extension of a concept from a subset of its examples (the entire extension is assumed to be directly observed), but to decide if the concept admits a given syntactical description.

It turns out that deciding whether the tightest approximation exactly describes a given concept, even when the concept is of small size, might require insurmountable computation; a problem not normally addressed in the literature on PAC learning.

The differences between learnability and identifiability can be well demonstrated using our previous example of the class C_k of k-CNF theories. We have established earlier that while C_k is not identifiable relative to the class C' of all relations, it is nevertheless strongly identifiable relative to $C' = C_k$. By comparison the class C_k is known to be polynomially learnable [Valiant, 1984] since, given a collection of instances I of $M(\varphi)$, one can find in polynomial time the tightest k-CNF expression that contains I (see section 3.1). The fact that C_k is not identifiable is not too disturbing in PAC learning tasks, because there we assume that the examples must be drawn from some k-CNF theory, so in the long run, the tightest k-CNF approximation to φ will eventually coincide with the theory from which φ is drawn. However, non-identifiability could be very disturbing if the possibility exists that the examples are taken from a theory outside C_k. In this case the tightest k-CNF theory consistent with the examples might lead to substantial (one-sided) errors.

In general, if we set $C' = C$, then, if C is learnable, it must also be strongly identifiable, because condition (1) is satisfied automatically, and the learnability requirement of zero error on negative examples is equivalent to (3). (Note that since the learner is entitled to observe the entire concept, the PAC requirement of limited error plays no role in identifiability tasks.) However, there are concept classes that are identifiable but not learnable under the condition $C' = C$, a simple example of which is the class of constraint trees. This class is not learnable because it is not closed under intersection, still, it has been shown to be identifiable [Meiri *et al.*, 1990; Dechter, 1990; Dechter and Pearl, 1991]. The same applies to star-structured networks. On the other hand, chains and k-trees are not identifiable [Dechter and Pearl, 1991].

3 Identifying Horn theories

In general, determining whether a given query formula follows from a given CNF formula is intractable. However, when the latter contains only Horn clauses the problem can be solved in linear time [Dowling and Gallier, 1984]. Moreover, experience with logic programming and databases suggests that humans find it natural to communicate knowledge in terms of Horn expressions. Thus, it would be useful to determine whether a given set of observations (the data ρ) can be described as a Horn theory.

The tractability of Horn theories stems not from the topology of the interactions among their clauses but, rather, from the syntactic restriction imposed on each individual clause. However, there are several impediments to the prospects of identifying general Horn theories. First, Selman and Kautz have shown that finding a tightest Horn approximation to a given CNF formula is NP-hard [Selman and Kautz, 1991]. All indications are that starting with a given relation does not make this task any easier. Second, Selman and Kautz also observed that some CNF theories can be converted into Horn expressions only after invoking exponentially many clauses (in the size of the source theory). In such cases it will be futile to use the Horn theory instead of the observations themselves. The more practical question to ask then is whether a given relation can be described as a Horn theory of a reasonable size. To that end, we first analyze the identifiability of k-Horn formulas, namely, Horn formulas in which every clause contains at most k literals, and then extend the results to Horn theories of limited overall size. We start by analyzing general CNF formulas parameterized by their scheme.

3.1 Canonical and Maximal Formulas

Paralleling the multi-valued case, we will first extend the auxiliary notions of *projection network* and *minimal network* to those of *projection formula* and *maximal formula.*

Definition 5 *Let ρ be a bi-valued relation over $X = X_1, ..., X_n$. We define*

$$canonical(\rho) = \{(\sim x_1 \vee \sim x_2 \vee ... \vee \sim x_n) | (x_1, x_2, ..., x_n) \notin \rho\} \quad (7)$$

Example 3 *Let $\rho(P, Q, R) = \{(100), (010), (001)\}$. Then, $canonical(\rho) = \{(\neg P \vee \neg Q \vee R), (P \vee \neg Q \vee \neg R), (\neg P \vee Q \vee \neg R), (P \vee Q \vee R), (\neg P \vee \neg Q \vee \neg R\}$.*

Similarly,

Definition 6 *Given a constraint network $N = \{\rho_1, ..., \rho_t\}$, we define $canonical(N)$ as the formula generated by collecting the canonical formulas of every constituent relation in N. Namely,*

$$canonical(N) = \cup\{canonical(\rho_i) | \rho_i \in N\}. \quad (8)$$

Clearly, $M(canonical(\rho)) = \rho$, and $M(canonical(N)) = rel(N)$.

We are now ready to extend the notion of projection network to a projection formula:

Definition 7 *Given a relation ρ and a scheme S, the* **projection formula** *of ρ w.r.t S, denoted $\Gamma_S(\rho)$, is given by:*

$$\Gamma_S(\rho) = canonical(\Pi_S(\rho)). \quad (9)$$

Theorem 1 *Let F_S be the class of CNF formulas having scheme S. The formula $\Gamma_S(\rho)$ is a tightest approximation of ρ relative to F_S.*

Paralleling the notion of minimal networks in multi-valued relations, we will now show that among all formulas φ in F_S that are equivalent to $\Gamma_S(\rho)$, $\Gamma_S(\rho)$ is **maximal** w.r.t. the partial order $\subseteq$ defined by set inclusion (of clauses). Clearly the class F_S is closed under union. The next theorem proves that among all equivalent formulas in F_S, $\Gamma_S(\rho)$ is the unique maximal formula.

Theorem 2 *Let $\varphi, \tau \in F_S$ and let $\rho = M(\varphi)$, then*

1. $\varphi \approx \tau \Longrightarrow \varphi \cup \tau \approx \varphi$
2. *There exists a unique maximal (w.r.t $\subseteq$) formula μ_S representing ρ given by $\mu_S = \Gamma_S(\rho)$.*

A clause that contains another is clearly redundant, hence we prefer to consider formulas in **reduced** form:

Definition 8 *A formula φ is* **reduced** *if none of its clauses contains another. The formula obtained after eliminating clause subsumption from φ is denoted reduced(φ).*

Theorem 3 *Let μ_S be a maximal formula of some relation, then reduced(μ_S) contains all and only the prime-implicants of μ_S that are restricted to the subsets in S.*

3.2 k-Horn formulas

We now restrict our attention to k-Horn formulas and their identifiability. We will first present a tractable algorithm for generating the maximal tightest k-Horn approximation to a given relation, followed by a tractable test for exactness.

Let S^{*k} denote the set of all subsets of X of size k or less. Our algorithm can be stated as follows: Given a relation ρ on n variables and a constant k, generate the formula $\Gamma_{S^{*k}}(\rho)$ and throw away all non-Horn clauses. We claim that the resulting Horn theory is a tightest k-Horn approximation of ρ. Since, as we will show, this is also the longest form of the tightest approximation, we then generate its equivalent reduced version. To test if the resulting Horn theory represents ρ exactly, we enumerate its models and test that no one lies outside ρ. A formal justification to this process is given in the following paragraphs.

Given a formula φ, we denote by $Horn(\varphi)$ the formula resulting from eliminating all non-Horn clauses from φ.

Theorem 4 *Let ρ be an n-ary bi-valued relation, k a constant, $\pi = \Gamma_{S^{*k}}(\rho)$ and $\eta = Horn(\pi)$. Let H_k be the family of k-Horn formulas, then,*

1. *η is a tightest k-Horn approximation of π.*
2. *η is maximal w.r.t. H_k.*
3. *Both η and reduced(η) are tightest k-Horn approximations of ρ.*
4. *if $M(\eta) \supset \rho$, no k-Horn formula describes ρ.*
5. *reduced(η) equals the set of all k-Horn prime-implicants of η.*

Theorem 4 implies that the algorithm given below which generates the formula $reduced(Horn(\Gamma_{S^{*k}}(\rho)))$, is guaranteed to return a tightest k-Horn approximation of ρ. The algorithm also returns a statement as to whether the formula found is an exact representation of ρ.

Algorithm Horn-generation(ρ, k)
Input: a relation $\rho(X_1, ..., X_n)$ and an integer k.
Output: A k-Horn formula describing ρ or a k-Horn tightest approximation of ρ.

1. begin
2. generate $\pi \Longleftarrow \Gamma_{S^{*k}}(\rho)$ (by projecting ρ on all subsets and performing the canonical transformation)
3. Let $\mu \Longleftarrow Horn(\pi)$ (by eliminating all non-Horn clauses from π).
4. $\eta \Longleftarrow reduced(\mu)$. (by eliminating subsumptions)
5. Sequentially enumerate the models of η, $\{m_1, m_2, ...\}$, using the method in [Dechter and Itai, 1991], and
   - If for some $i \leq |\rho|$, $m_i \notin \rho$, or if $M(\eta)$ contains more than $|\rho|$ elements, then return: "η is a tightest k-Horn approximation." else, return: "η describes ρ".
6. end.

In [Dechter and Itai, 1991] we showed that the models of a Horn formula can be enumerated in time linear in the number of models and the size of the formula. However, in the above algorithm we do not need to compute more then $|\rho|$ models, thus this computation is bounded by $|\rho|$.

To summarize:

Theorem 5 *Algorithm Horn-generation provides a tightest k-Horn approximation of an arbitrary relation ρ. Moreover, this approximation equals the k-Horn prime-implicants of ρ.* □

Example 4 *Consider again the relation*

P	Q	R
1	0	0
0	1	0
0	0	1

and let $k = 2$. We have

$\Pi_{S^{*2}}(\rho) =$

P	Q
1	0
0	1
0	0

P	R
1	0
0	1
0	0

R	Q
1	0
0	1
0	0

and $P = \{0,1\}$, $Q = \{0,1\}$, $R = \{0,1\}$. When applying the canonical transformation to each of these relations we get the (already reduced) formula:

$$\Gamma_{S^{*2}}(\rho) = \{(\neg P \vee \neg Q), (\neg P \vee \neg R), ((\neg R \vee \neg Q)\}.$$

Since this is a Horn formula we do not throw clauses away. Computing the number of models of this theory yields 4 models (there is an additional (0,0,0) tuple), thus we conclude that the formula is a tightest 2-Horn approximation of ρ, and that ρ is not 2-Horn identifiable. If we generate the 3-Horn approximation for ρ we get the same formula. (The reason being that in this case, the 2-Horn approximation already contains all its Horn-prime implicants.) Going through the Horn-generation *algorithm, step 2 yields:*

$$\Gamma_{S^{*3}}(\rho) = \{(\neg P \vee \neg Q \vee R), (P \vee \neg Q \vee \neg R), (\neg P \vee Q \vee$$
$$(P \vee Q \vee R), (\neg P \vee \neg Q \neg R), (\neg P \vee \neg Q), (\neg P \vee \neg R), \neg R \vee -$$

Step 3 eliminates the only non-Horn clause: $(P \vee Q \vee R)$ and the result of further eliminating subsumptions is the same formula:

$$Horn(reduced(\Gamma_{S^{*3}}(\rho))) = \{(\neg P \vee \neg Q), (\neg P \vee \neg R), \neg R \vee \neg Q$$

(10)

This suggests an *anytime* variation of our algorithm. Instead of applying the algorithm to all subsets of size k, we first apply the algorithm to subsets of size 2, then add the result of processing subsets of size 3, and so on, until we get a satisfying approximation. The next theorem assesses the complexity of our transformation and the size of its resulting Horn theory.

Theorem 6 *(complexity)*

1. *The length (number of clauses) of* $reduced(Horn(\Gamma_{S^{*k}}(\rho)))$ *is* $O(kn^{k+1})$.
2. *The complexity of* Horn-generation(ρ, k) *is* $O(n^k((k+1)|\rho| + 2^k))$.

Another important variant of the method described above is its *on-line* version, which is useful for stream processing. Assume the tuples of ρ are not available all at once, but are obtained sequentially as a stream of observations, normally containing many repetitions. In this case it might be advantageous to store a parsimonious theory of past data, rather than the data itself, and to update the theory incrementally whenever an observation arrives that contradicts the theory.

Assume we are given a theory h which is a tightest k-Horn approximation of all past data, ρ, and a new tuple t arrives that contradicts h. In principle, updating h requires finding a tightest k-Horn theory that agrees with $\rho \cup \{t\}$. but, since ρ is no longer available, the best we can do is to find a tightest k-Horn approximation of $M(h) \cup \{t\}$. Fortunately, since H_k is closed under intersection, we are guranteed that the two approximations are equivalent, namely, no information is lost by storing h instead of the exact stream of past observations.

The next theorem states that updating h can be done in polynomial time. Although each update may, in the worst case require as many as $O(n^{k+1})$ steps, it is nevertheless polynomial, and is more efficient than approximating $\rho \cup \{t\}$ from scratch when the size of ρ is exponential in n.

Theorem 7 *Incremental updating of best k-Horn approximations takes* $O(n^{k+1})$ *steps per update.*

Clearly, the facility for incremental on-line updating would be useful only when the size of ρ is the main factor that limits our ability to find a useful description of the data.

3.3 Extensions to general Horn formulas

A recent algorithm by [Angluin *et al.*, 1990] permits us to extend the results of the last section to the identification of Horn theories of size $q(n)$, for any fixed polynomial q.[2] The algorithm of Angluin et al. exactly learns Horn theories from equivalence queries and membership queries. An equivalence query is a conjectured Horn theory, and the response by the teacher is a counterexample to the correctness of the conjectured Horn theory (i.e. an assignment that satisfies the correct theory but not the conjectured theory, or vice versa). In the case that there are no counterexamples, then the learning algorithm has succeeded in identifying the correct theory. Membership queries allow the algorithm to ask if a given assignment satisfies the target (i.e., correct) Horn theory, and it is answered yes or no by the teacher.

To be able to answer equivalence queries in polynomial time, Angluin et al. assumed that the target theory is Horn (in general, testing equivalence of two given theories is intractable). If we are given a relation ρ, then we can answer equivalence queries and provide counterexamples in polynomial time even when ρ is non-Horn. Given a conjectured Horn theory H, we first check that every tuple of ρ satisfies H. If not, we return the unsatisfying tuple as a counterexample. Otherwise, $M(H)$ contains ρ, and we then determine whether or not $M(H) = \rho$ by the polynomial enumeration method of [Dechter and Itai, 1991].

Thus, since we can polynomially answer the two basic queries of Angluin's learning algorithm, the algorithm must output an exact Horn representation of ρ if one exists. To determine whether or not one exists of size at most $q(n)$, we can run the algorithm for $t(n, q(n)$ time, where $t(n, k)$ is the time needed by the algorithm to exactly learn a Horn theory of size k over n variables. If the algorithm succeeds in exactly learning ρ within $t(n, q(n))$ time, then there clearly is a Horn theory for ρ of size at most $q(n)$. Otherwise, there is not. Of course, in this case the algorithm does not supply a tightest Horn approximation, and the strong identifiability of $q(n)$-Horn theories remains an open problem.

4 Conclusions

This paper summarizes several investigations into the prospects of identifying meaningful structures in empirical data. The central theme is to identify a computationally attractive description, in cases where the observed data possess such a description and a best approximate description otherwise. This feasibility of performing this task in reasonable time has been given a formal definition through the notion of identifiability, which is normally weaker (if $C' = C$) than that of learnability.

In a related paper [Dechter and Pearl, 1991] we have explored more generally the decomposition of data into a given scheme of smaller relations, as illustrated in Section 2.3. It can be shown that, whereas a best approximation can be found, it is only in cases where the scheme is intrinsically tractable (e.g., 2-CFN) that we can (tractably) decide if the resulting approximation constitutes an exact representation of the data. The decomposition of data into a structure taken from a class of schemes turned out to be a harder task, intractable even in cases where each individual member of the class is tractable. The class of tree-structured schemes is an exception. Here it was shown that an effective procedure exists for determining whether a given relation is decomposable into a tree of binary relations and, if the answer is positive, identifying the topology of such a tree. The procedure runs in time proportional to the size of the relation, but it is still an open question whether it provides the best tree-structured approximation in case the answer is negative.

Focusing on bi-valued data, this paper has explored the identification of descriptions whose tractability stems

[2] This possibility was brought to our attention by an anonymous reviewer of [Dechter and Pearl, 1991].

from syntactical rather than structural features. In particular, we showed that k-Horn theories can be identified in polynomial time, when k is bounded. Finally, the paper presents both any-time and on-line algorithms for identifying Horn theories.

An important issue that was not dealt in this paper is assessing the goodness of the approximations provided by k-Horn theories. Another question is the feasibility of constructing both an upper bound and a lower bound approximations of ρ, in the manner discussed in [Selman and Kautz, 1991] and also in [Dechter, 1990]. Finally, we should mention that the methods presented in this paper will also handle partial observations, namely, observations of truncated tuples of ρ.

5 Acknowledgments

We thank Itay Meiri and Amir Weinshtain for useful discussions.

References

[Angluin *et al.*, 1990] D. Angluin, M. Frazier, and L. Pitt. Leaving conjunctions of horn clauses. In *Proceedings of the 31st Annual Symposium on Foundation of Computer Science, Volume I*, St. Louis, MS, October 1990. IEEE Computer Society Press.

[Chow and Liu, 1968] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Info. Theory*, IT-(14):462–467, 1968.

[Dechter and Itai, 1991] R. Dechter and A. Itai. The complexity of finding all solutions. Technical report, University of California at Irvine, 1991.

[Dechter and Pearl, 1991] R. Dechter and J. Pearl. Structure identification in relational data. Technical Report R-172, Cognitive Systems Laboratory, UCLA, 1991. Forthcoming *Artificial Intelligence.*

[Dechter, 1990] R. Dechter. Decomposing a relation into a tree of binary relations. *Journal of Computer and System Sciences*, 41(Special Issue on the Theory of Relational Databases):2–24, 1990.

[Dowling and Gallier, 1984] W. F. Dowling and J. H. Gallier. Linear time algorithms for testing the satisfiability of propositional horn formula. *Journal of Logic Programming*, 3:267–284, 1984.

[Maier, 1983] D. Maier. *The Theory of Relational Databases.* Computer Science Press, Rockville, Maryland, 1983.

[Meiri *et al.*, 1990] I. Meiri, R. Dechter, and J. Pearl. Tree decomposition with applications to constraint processing. In *Proceedings of the the American Association of Artificial Intelligence (AAAI-90)*, pages 10–16, Boston, MA, 1990.

[Natarajan, 1987] B.K. Natarajan. On learning boolean functions. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computation*, pages 296–304, 1987.

[Pearl and Verma, 1991] J. Pearl and T. Verma. A theory of inferred causation. In J. A. Allen, R. Fikes, and E. Sandewall, editors, *In Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 441–452, San Mateo, CA, 1991. Morgan Kaufmann.

[Selman and Kautz, 1991] B. Selman and H. Kautz. Knowledge compilation using horn approximation. In *In Proceedings of AAAI-91*, Aneheim, CA, 1991.

[Valiant, 1984] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Inference in Inheritance Networks using Propositional Logic and Constraint Networks Techniques

Rachel Ben-Eliyahu
rachel@cs.ucla.edu
Cognitive Systems Laboratory
Computer Science Department
University of California
Los Angeles, California 90024

Rina Dechter
dechter@ics.uci.edu
Information & Computer Science
University of California
Irvine, California 92717

Abstract

This paper focuses on *network default theories*. Etherington [Etherington, 1987] has established a correspondence between inheritance networks with exceptions and a subset of Reiter's default logic called network default theories, thus providing a formal semantics and a notion of correct inference for such networks. We show that any such propositional network default theory can be compiled in polynomial time into a classical propositional theory such that the set of models of the latter coincides with the set of extensions of the former. We then show how constraint satisfaction techniques can be used to compute extensions and to identify tractable network default theories. For any propositional network theory, our algorithms compute all its extensions and verifies if a given conclusion is in one or all extensions.

1 Introduction

Research in multiple inheritance networks has focused on two main issues: developing fast algorithms that will operate on the network links to produce conclusions that match our intuition, and providing formal semantics for such networks. Clearly, the second is crucially important for adequate evaluation of the correctness of the first.

Etherington [Etherington, 1987] had approached the semantic issue by formalizing inheritance networks, called *network default theories*, within Reiter's default logic. While his framework has been criticized for demanding all exceptions be listed explicitly, his approach is still valuable in that it embeds the notion of inheritance within this general and widely studied framework of default logic.

Our paper focuses on the computational aspects of such network theories. We first present a necessary and sufficient condition for their coherence, namely, for deciding whether or not they have an extension. Then, using constraint satisfaction techniques, we present effective schemes for computing the extensions for *any* such network. In contrast, Etherington's procedure is only applicable to a subclass of networks theories called "ordered network theories". Moreover, the complexity of our schemes is related to the sparseness of the networks, as captured by the parameter of *induced width*.

The approach leading to these results has already been applied to the subclass of propositional disjunction-free semi-normal default theories [Ben-Eliyahu and Dechter, 1991a]. We have shown there that any such default theory can be compiled in polynomial time into a propositional theory, such that each of its models corresponds to an extension of the default theory. Constraint network techniques are then applied to compute extensions and to identify, analyze and solve tractable subclasses of this default logic. A generalization of this approach to network theories requires allowing size-two disjunctions in the default theory.

Our results pave the way for applying constraint networks techniques to logic programming as well since it has been shown that there is a one-to-one correspondence between stable models of logic programs and extensions of each of their default interpretations [Gelfond and Lifschitz, 1990]. Elkan [Elkan, 1990] has also shown that stable models of a logic program with no classical negation can be represented as models of propositional logic.

The paper is organized as follows: in section 2 we briefly introduce default logic and inheritance networks. In section 3 we describe how tasks of default theories are mapped into equivalent tasks in propositional logic. This mapping is exploited in section 4 where we present new procedures for query processing and identify tractable classes using constraint networks techniques. Section 5 provides concluding remarks. Due to space considerations all proofs are omitted. For more details see [Ben-Eliyahu and Dechter, 1991b].

2 Default logic and inheritance networks

2.1 Reiter's default logic

Following is a brief introduction to Reiter's default logic [Reiter, 1980]. Let $\mathcal{L}$ be a first order language. A *default theory* is a pair $\Delta = (D, W)$, where D is a set of defaults and W is a set of closed wffs (well formed formulas) in $\mathcal{L}$. A *default* is a rule of the form $\alpha : \beta_1, ..., \beta_n/\gamma$, where $\alpha, \beta_1, ...\beta_n$ and γ are formulas in $\mathcal{L}$. The intuition behind a default can be: if I believe α, and I have no reason to

believe that one of the β_i is false, then I can believe γ. A default $\alpha : \beta/\gamma$ is *normal* if $\gamma = \beta$. A default is *semi-normal* if it is in the form $\alpha : \beta \wedge \gamma/\gamma$. A default theory is *closed* if all the first order formulas in D and W are closed.

The set of defaults D induces an *extension* on W. Intuitively, an extension is a maximal set of formulas that can be deduced from W using the defaults in D. Let $Th(E)$ denote the logical closure of E in $\mathcal{L}$. We use the following definition of an extension:

Definition 2.1 *([Reiter, 1980],theorem 2.1) Let $E \subseteq \mathcal{L}$ be a set of closed wffs, and let $\Delta = (D, W)$ be a closed default theory. Define*[1]

- $E_0 = W$
- *For $i \geq 0$ $E_{i+1} = Th(E_i) \bigcup \{\gamma | \alpha : \beta_1, ..., \beta_n/\gamma \in D$ where $\alpha \in E_i$ and $\neg\beta_1, ...\neg\beta_n \notin E\}$.*

E is an extension for Δ iff for some ordering $E = \bigcup_{i=0}^{\infty} E_i$ □

Most tasks on a default theory Δ can be formulated using one of the following queries:

Coherence: Does Δ have an extension ? If so, find one.

Set-Membership: Given a set of formulas S, Is S contained in some extension of Δ?

Set-Entailment: Given a set of formulas S, Is S contained in every extension of Δ?

In this paper we show how, for a subclass called "network default theories", the above queries can be reduced to propositional satisfiability.

2.2 Inheritance networks and network default theories

The following brief introduction is adopted from [Etherington, 1987] and [Touretzky, 1984].

Inheritance networks are a knowledge representation scheme in which the knowledge is organized in a taxonomic hierarchy, thus allowing representational compactness. If many individuals share a group of common properties, an abstraction of those properties is created, and all those individuals can "inherit" from that abstraction. Inheritance from multiple classes is also allowed.

Usually, the inheritance network is a directed graph whose nodes represent individuals and abstractions ("classes"), and whose arcs denote relations between those nodes. The most common relations are "IS-A" and "ISN'T-A".

Consider the following information:

- Mammals are warm-blooded.
- Dolphins are mammals.
- Flipper is a dolphin.

This information can be encoded in the inheritance network shown in figure 1 (where a solid arrow represents an "IS-A" relation). A reasonable conclusion would be that Flipper is warm-blooded.

When exceptions to inheritance are allowed, the inference in those systems becomes non-monotonic, namely, conclusions might change in light of new evidence. Suppose we start with the following set of axioms:

[1]Note the appearance of E in the formula for E_{i+1}.

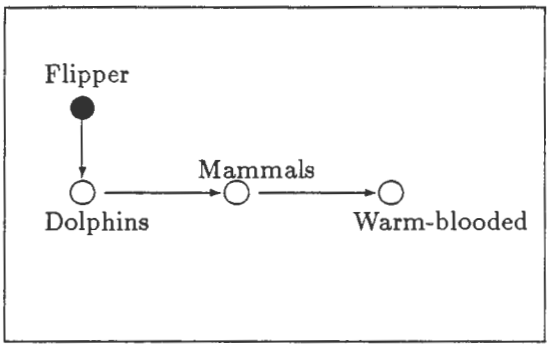


Figure 1: An inheritance network with no exceptions

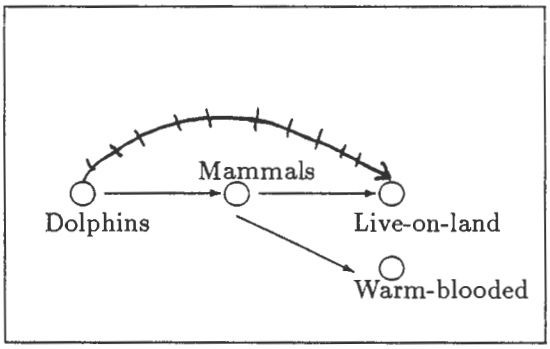


Figure 2: An inheritance network with exceptions

- Mammals are warm-blooded.
- Mammals live on land.
- Dolphins are mammals.
- Dolphins do not live on land.

This is an example of an inheritance network with exception: dolphins are mammals who live in the water. The network in figure 2 represents this knowledge ("canceled" arrows denote "ISN'T-A" relation). Given the information that Flipper is a mammal, we will conclude that he lives on land, but the additional evidence that he is a dolphin will force us to retract that conclusion and adopt the belief that he does not live on land[2].

Etherington [Etherington, 1987] proposed a subclass of default theories called "network default theories" (in short, "network theories") as suitable to provide formal semantics and a notion of sound inference for those networks:

Definition 2.2 *(*Network default theory*) [Etherington, 1987] A default theory Δ is a* network default theory *iff it satisfies the following conditions:*

- *W contains only:*
  - *literals (i.e atomic formulae or their negations), and*
  - *disjuncts of the form $(\alpha \vee \beta)$ where α and β are literals.*
- *D contains only normal and seminormal defaults of the form: $\alpha : \beta/\beta$ or $\alpha : \beta \wedge \gamma_1 \wedge ... \wedge \gamma_n/\beta$ where α, β and γ_i are literals.* □

[2]This conclusion is supported by the convention that features of a subclass override those of a super-class.

Etherington suggests a way to formalize inheritance relations in network theories. His translation is as follows:

Strict IS-A: "A's are always B's". Etherington suggests translating this to the first-order formula $\forall x.A(x) \longrightarrow B(x)$. Since we restrict our treatment to propositional theories, we will translate this link to the propositional rule schema $A(x) \longrightarrow B(x)$.

Membership: "The individual a belongs to the class A". This is represented by the fact $A(a)$ (which denotes here a propositional literal).

Strict ISN'T-A: "A's are never B's". Etherington translates this to the first-order formula $\forall x.A(x) \longrightarrow \neg B(x)$. We will translate this link to the propositional rule schema $A(x) \longrightarrow \neg B(x)$.

Nonmembership: "The individual a does not belong to the class A". This is represented by the fact $\neg A(a)$.

Default IS-A: "A's are normally B's, but exceptions are allowed". This can be represented by the default rule schema $A(x) : B(x)/B(x)$.

Default ISN'T-A: "Normally A's are not B's, but exceptions are allowed". This can be represented by the default rule schema $A(x) : \neg B(x)/\neg B(x)$.

Exception: "Normally A's are (not) B's, unless they have at least one of the properties $C_1, ..., C_n$". This translates to the default rule schema $A(x) : B(x) \wedge \neg C_1(x) \wedge ... \wedge \neg C_n(x)/B(x)$
$(A(x) : \neg B(x) \wedge \neg C_1(x) \wedge ... \wedge \neg C_n(x)/\neg B(x)$)

Example 2.3 *The inheritance network in figure 2 will be translated to the following network theory:*

$$D = \{\frac{Mammal(x):Lives\text{-}on\text{-}land(x) \wedge \neg Dolphine(x)}{Lives\text{-}on\text{-}land(x)}, \frac{Dolphine(x):\neg Lives\text{-}on\text{-}land(x)}{\neg Lives\text{-}on\text{-}land(x)}\}$$

$$W = \{\ Dolphine(x) \longrightarrow Mammal(x), Mammal(x) \longrightarrow Warm\text{-}blooded(x)\ \}$$ □

An extension of a network theory then corresponds to a set of coherent conclusions one could draw from the inheritance network it represents. Thus all the queries defined above (coherence, set-membership, set-entailment) are still very relevant when dealing with network theories. Etherington has a nondeterministic procedure to compute an extension of a default theory. If the theory is what he calls an *ordered* network theory, then his procedure is guaranteed to produce an extension.

In the sequel we will show a procedure that computes all extensions for *any propositional* network theory. In fact, we deal with a superclass in which the prerequisite of a default is a conjunction of literals rather than just a single literal. We will assume, w.l.g., that W is consistent, since when W is inconsistent, the extension is the inconsistent one. We also assume w.l.g. that each default has a single literal as a consequent.

3 Definitions and preliminaries

We denote propositional symbols by upper case letters $P, Q, R...$, propositional literals (i.e. $P, \neg P$) by lower case letters $p, q, r...$, clauses by $c_1, c_2, ...$ The number of literals in the clause c is denoted by $|c|$.

The operator $\sim$ over literals is defined as follows: If $p = \neg Q$, $\sim p = Q$, If $p = Q$ then $\sim p = \neg Q$. If $\delta = \alpha : \beta/\gamma$ is a default, we define $\text{pre}(\delta) = \alpha$, $\text{just}(\delta) = \beta$ and $\text{concl}(\delta) = \gamma$.

Given a set of formulas S and a formula ω, $S \vdash \omega$ means that ω is provable from premises S, and $S \models \omega$ means that S entails ω - i.e. that every model of S satisfies ω as well. For propositional formulas, $S \vdash \omega$ iff $S \models \omega$, hence we will use these notations interchangeably.

The *logical closure* of a set of formulas S is the set $\{\omega | S \vdash \omega\}$. We denote by $Th(S)$ the *logical closure* of a set of formulas S.

An extension of a default theory is a logically closed set of formulas. How do we compute the logical closure of a set of clauses? Since the logical closure is an infinite set, we will not be able to compute the closure in a finite time. However, if the initial set of clauses is finite, we can compute a set which will represent the logical closure using the notion of *prime implicants* as presented by Reiter and de Kleer [Reiter and de Kleer, 1987]:

Definition 3.1 *A prime implicant of a set S of clauses is a clause c such that*

1. *$S \models c$, and*
2. *there is no proper subset c' of c such that $S \models c'$.*

Given a set of formulas S, S^+ will denote the set of its prime implicants. As Reiter and de Kleer note, a brute force method of computing S^+ is to repeatedly resolve pairs of clauses of S, add the resolvents to S, and delete subsumed clauses, until a fixed point is reached[3]. There are some improvements to that method, but it is clear that the general problem is NP-Hard since it also solves satisfiability. Nevertheless, for size-2 clauses the prime implicants can be computed in polynomial time since a resolvent of two clauses of size ≤ 2 is also of size ≤ 2.

The following proposition suggests that for network theories it is enough to consider extensions of a network theory containing clauses of size one or two only:

Proposition 3.2 *Let E^* be an extension of a network theory, and let $E' = \{c | c \in E^*, |c| \leq 2\}$. Then E' contains all prime implicants of E^*.* □

We say that a set of clauses E *satisfies the preconditions of* δ if $\text{pre}(\delta) \in \text{Th}(E)$ and the negation of $\text{just}(\delta)$ is not in $\text{Th}(E)$. We say that E *satisfies a default* δ if it does not satisfy the preconditions of δ or else, it satisfies its preconditions and $\text{Th}(E)$ contains its conclusion.

A *proof of a clause* c w.r.t. a given set of clauses E and a given network theory $\Delta = (D, W)$ is a sequence of rules $\delta_1, ..., \delta_n$, $n \geq 0$, such that the following three conditions hold:

1. $c \in \text{Th}(W \bigcup \{\text{concl}(\delta_1), ..., \text{concl}(\delta_n)\})$.
2. For all $1 \leq i \leq n$, the negation of $\text{just}(\delta_i)$ is not in $\text{Th}(E)$.
3. For all $1 \leq i \leq n$, $\text{pre}(\delta_i)$ is a subset of $\text{Th}(W \bigcup \{\text{concl}(\delta_1), ..., \text{concl}(\delta_{i-1})\})$.

[3] It is clear that this method will not generate all the tautologies, but it is easy to handle this exception.

The following lemma is instrumental throughout the paper:

Lemma 3.3 *$Th(E)$ is an extension of a network theory Δ iff $Th(E)$is a logical closure of a set of clauses E that satisfies:*

1. *$W \subseteq E$*
2. *E satisfies each rule in D.*
3. *For each clause $c \in E$, there is a proof of c in E .* □

We define the *dependency graph* $G_{(D,W)}$ of a network theory Δ to be a directed graph constructed as follows: Each literal p appearing in D or in W is associated with a node, and an edge is directed from p to r iff there is a default rule where p appears in its prerequisite and r is its consequent or there is a clause $p \longrightarrow r$ in W. An *acyclic network theory* is one whose dependency graph is acyclic, a property that can be tested linearly.

4 Compiling a network theory into a propositional theory

In this section we show how we can compile a given network theory Δ into a propositional theory $\mathcal{P}_\Delta$ such that $\mathcal{P}_\Delta$ has a model iff Δ has an extension, and vice-versa, every model of $\mathcal{P}_\Delta$ has a corresponding extension for Δ.

The common approach for building an extension, (used by Etherington [Etherington, 1987], Kautz and Selman [Kautz and Selman, 1991], and others), is to increment W using rules from D. We make a declarative account of this process by formulating the conditions of lemma 3.3 as a set of constraints that the default theory impose on the set of its extensions. This frees us from worrying about ordering, however, it requires adding a constraint guaranteeing that if a formula is in the extension, then it has a non-circular proof. To enforce this restriction, we associate an *index variable* with each literal in the transformed language, and require that p is in the extension only if it is the consequent of a rule whose prerequisite's indexes are smaller. Elkan [Elkan, 1990] used the same technique to insure that the justifications supporting a node in a TMS are noncircular.

Let $\#p$ stand for the "index associated with p", and let k be its number of values. These "multi-valued variables" (as opposed to propositional variables which are bi-valued) can be expressed in propositional logic using additional $O(k^2)$ clauses and literals (see [Ben-Eliyahu and Dechter, 1991b]). For simplicity, however, we will use the multi-variable notations, viewing them as abbreviations to their propositional counterparts.

Let $\mathcal{L}$ be the underlying propositional language of Δ. For each propositional symbol in $\mathcal{L}$, we define two propositional symbols, I_P and $I_{\neg P}$. For each pair of literals p and q in $\mathcal{L}$, we define the symbol $I_{p \vee q}$. We get a new set of symbols : $\mathcal{L}' = \{I_P, I_{\neg P} | P \in \mathcal{L}\} \bigcup \{I_{p \vee q} | p, q \in \mathcal{L}\}$. Intuitively, I_P stands for "P is in the extension", $I_{\neg P}$ stands for "$\neg P$ is in the extension", and $I_{p \vee q}$ means that "$p \vee q$ is in the extension". For notational convenience I_p and $I_{p \vee p}$ will stand for the same propositional letter (same for $I_{\sim p \vee q}$ and $I_{p \longrightarrow q}$).

Procedure translate-1(Δ)

1. Compute W^+, the set of prime implicants of W.
2. For each $c \in W^+$ put I_c into $\mathcal{P}_\Delta$.
3. For each $p \longrightarrow q$ in W add $I_p \longrightarrow I_q$ into $\mathcal{P}_\Delta$.
4. For each $\alpha : \beta / p \in D$, add $in(\alpha) \wedge cons(\beta) \longrightarrow I_p$ to $\mathcal{P}_\Delta$.
5. For each $p \notin W^+$ do the following :
   Let $C_p = \{[in(q_1 \wedge q_2 ... \wedge q_n) \wedge cons(\beta)] \wedge [\#q_1 < \#p] \wedge ... \wedge [\#q_n < \#p] \mid \exists \delta \in D$ such that $\delta = q_1 \wedge q_2 ... \wedge q_n : \beta / p \}$.
   Let $L_p = \{[in(q) \wedge [\#q < \#p]] \mid q \longrightarrow p \in W^+\}$.
   Let $S_p = C_p \bigcup L_p$.
   If S_p is not empty then add to $\mathcal{P}_\Delta$ the formula $I_p \longrightarrow [\vee_{\alpha \in S_p} \alpha]$.
   Else, if $S_p = \emptyset$ add $\neg I_p$ to $\mathcal{P}_\Delta$.
6. For each $p \vee q \notin W^+$, $p \neq q$, add $\neg I_{p \vee q}$ into $\mathcal{P}_\Delta$.

Figure 3: Algorithm to compile a network theory into a propositional theory

To further simplify the notation we use the notions of $in(\omega)$ and $cons(\omega)$ that stand for "ω is in the extension". and "ω is consistent with the extension", respectively. Formally, $in(\omega)$ and $cons(\omega)$ are defined as follows:

- if $\omega = p$ then $in(\omega) = I_p$, $cons(\omega) = \neg I_{\sim p}$.
- if $\omega = p \vee q$ then $in(\omega) = I_{p \vee q}$.
- if $\omega = p_1 \wedge p_2 \wedge ... \wedge p_n$, then $in(\omega) = in(p_1) \wedge in(p_2) \wedge ... \wedge in(p_n)$, $cons(\omega) = \wedge_{i,j \in \{1,...,n\}} \neg I_{\sim p_i \vee \sim p_j}$. (Note that $p_1 \wedge ... \wedge p_n$ is "consistent with the extension" iff $\neg[p_1 \wedge ... \wedge p_n]$ is not in the extension iff (since all prime implicants are of size ≤ 2) for all i, j, $\sim p_i \vee \sim p_j$ is not in the extension.)

Procedure **translate-1** in figure 3 compiles any network theory over $\mathcal{L}$ into a propositional theory over $\mathcal{L}''$. This translation requires adding n index variables, n being the number of literals in $\mathcal{L}$, each having at most n values. Since expressing an *inequality* in propositional logic requires $O(n^2)$ clauses, and since there are at most n possible inequalities per default, the resulting size of this transformation is bounded by $O(|D|n^3)$ propositional sentences. Note also that the complexity of generating W^+ is at most $O(n^3)$.

The following theorems summarize the properties of our transformation. In all of them, $\mathcal{P}_\Delta$ is the set of sentences resulting from translating a given network theory Δ using translate-1.

Theorem 4.1 *Let Δ be a network theory. Suppose $\mathcal{P}_\Delta$ is satisfiable and θ is a model for $\mathcal{P}_\Delta$, and let $E = \{c | \theta(I_c) = \mathbf{true}\}$.*
Then:

1. *E contains all its prime implicants.*
2. *$Th(E)$ is an extension of Δ.* □

Theorem 4.2 *Let $Th(E)$ be an extension for Δ. Then there is a model θ for $\mathcal{P}_\Delta$ such that $\theta(I_c)$ = true iff $c \in Th(E)$ and $|c| \leq 2$.* □

The above two theorems suggest a necessary and sufficient condition for the coherence of a network theory:

Corollary 4.3 *A network theory Δ has an extension iff $\mathcal{P}_\Delta$ is satisfiable.* □

For the next corollaries, we define for each clause c a formula prime(c) as follows: if $c = p_1 \vee p_2 \vee ... \vee p_n$ prime(c) = $[\vee_{i,j \in \{1...n\}} I_{p_i \vee p_j}]$

Corollary 4.4 *A set of clauses, C, is contained in an extension of Δ iff there is a model for $\mathcal{P}_\Delta$ which satisfies the set $\{prime(c) | c \in C\}$.*

Corollary 4.5 *A clause c is in every extension of a network theory Δ iff $\mathcal{P}_\Delta \models prime(c)$.* □

These theorems suggest that we can first translate a given network theory Δ to $\mathcal{P}_\Delta$ and then answer queries as follows: to test if Δ has an extension, we test satisfiability of $\mathcal{P}_\Delta$; to see if a set S of clauses is a member in some extension, we test satisfiability of $\mathcal{P}_\Delta \bigcup \{\text{prime}(c) | c \in S\}$; and to see if S is included in every extension, we test if $\mathcal{P}_\Delta$ entails the formula $[\wedge_{c \in S} \text{prime}(c)]$.

Example 4.6

Consider again the network theory from example 2.3 together with the evidence that Flipper is a mammal (predicates are abbreviated by their initials; parameters are omitted since Flipper is the only individual):

$D = \{M : L \wedge \neg D / L,\ D : \neg L / \neg L\}$
$W = \{D \longrightarrow M, M \longrightarrow Wb, M\}$

This is an acyclic network theory, thus no indices are required. When translating Δ to $\mathcal{P}_\Delta$ we get:

$W^+ = W \bigcup \{Wb, D \longrightarrow Wb\}$
$\mathcal{P}_\Delta = \{$
following step 2:
$I_D \longrightarrow_M, I_M \longrightarrow_{Wb}, I_{Wb}, I_M, I_D \longrightarrow_{Wb}$
following step 3:
$I_D \longrightarrow I_M,\ I_M \longrightarrow I_{Wb},\ I_D \longrightarrow I_{Wb}$
following step 4:
$I_M \wedge \neg I_{\neg L} \wedge \neg I_D \wedge \neg I_{\neg L \vee D} \longrightarrow I_L,$
$I_D \wedge \neg I_L \longrightarrow I_{\neg L}$
following step 5:
$I_L \longrightarrow I_M \wedge \neg I_{\neg L} \wedge \neg I_D \wedge \neg I_{\neg L \vee D},$
$I_{\neg L} \longrightarrow I_D \wedge \neg I_L,\ \neg I_{\neg M},\ \neg I_D,\ \neg I_{\neg D},\ \neg I_{\neg Wb}$
following step 6:
$\{\neg I_{x \longrightarrow y} | x \longrightarrow y \notin \{D \longrightarrow M, M \longrightarrow Wb,$
$D \longrightarrow Wb\}\}$

This set of sentences has only one model in which all and only the following literals are true:

$$I_M, I_{Wb}, I_L, I_{D \longrightarrow M}, I_{M \longrightarrow Wb}, I_{D \longrightarrow Wb}$$

which correspond to the extension

$$Th(\{M, Wb, L, D \longrightarrow M, M \longrightarrow Wb\})$$

Example 4.7

Suppose we add the information that Flipper is a dolphin to what we knew in the previous example. This amounts to adding the proposition D to W. So we have to take $\neg I_D$ out of $\mathcal{P}_\Delta$ and add I_D to $\mathcal{P}_\Delta$. The model for $\mathcal{P}_\Delta$ is:

$$I_D, I_M, I_{Wb}, I_{\neg L}, I_{D \longrightarrow M}, I_{M \longrightarrow Wb}, I_{D \longrightarrow Wb}$$

which corresponds to the extension

$$Th(\{D, M, Wb, \neg L, D \longrightarrow M, M \longrightarrow Wb\})$$

which is the only extension.

4.1 An improved translation

Procedure translate-1 can be improved. If a prerequisite of a rule is not on a cycle with its consequent, we do not need to index them, nor enforce the partial order among their indices. Thus, we need indices only for literals which reside on cycles in the *dependency graph*. Furthermore, since we will never have to solve cyclicity between two literals that do not share a cycle, the range of the index variables is bounded by the maximum number of literals that share a common cycle. In fact, we show that the index variable's range can be bounded by the maximal length of an acyclic path in any *strongly connected component* in $G_{(D,W)}$. The strongly-connected components of a directed graph are a partition of its set of nodes such that for each subset C in the partition, and for each $x, y \in C$, there are directed paths from x to y and from y to x in G. This improvement is discussed in detail in [Ben-Eliyahu and Dechter, 1991b].

5 Tractable network default theories

Processing the network theory using our approach requires two steps: first, compile the default theory into a propositional theory, and then solve satisfiability. We have shown that the first step is tractable. The second step, however, is known to be NP-complete in general. In this section we show how propositional satisfiability can be regarded as a constraint satisfaction problem, and how techniques borrowed from that field can be used to solve satisfiability and to identify tractable subsets of propositional and network theories.

In general, constraint satisfaction techniques exploit the structure of the problem through the notion of a "constraint graph". For a propositional theory, the constraint graph (also called a "primal constraint graph") associates a node with each propositional letter and connects any two nodes whose associated letters appear in the same clause. Various graph parameters have been shown as crucially related to solving the satisfiability problem. These include the *induced width*, w^*, the *size of the cycle-cutset*, the *depth of a depth-first-search spanning tree* of this graph and the *size of the non-separable components*. It can be shown that the worse-case complexity of deciding consistency is polynomially bounded by any one of these parameters.

Since these parameters can be bounded easily by simple processing of the given graph, they can be used for assessing tractability ahead of time. For instance,

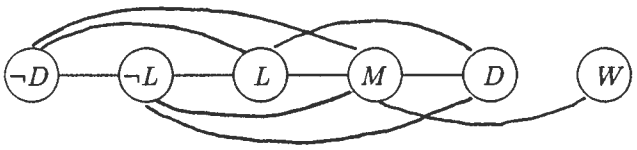


Figure 4: Interaction graph for the theory presented in example 4.6

when the constraint graph is a tree, satisfiability can be determined in linear time. In [Ben-Eliyahu and Dechter, 1991a] we have demonstrated the potential of this approach using one specific technique called *Tree-Clustering* [Dechter and Pearl, 1989], customized for solving propositional satisfiability, and emphasized its effectiveness for maintaining a default database. We have also characterized the tractability of the default theories as a function of the *induced width*[4] [5], w^*, of their *interaction graph*. We next generalize those results for network theories:

The *interaction graph* of a network theory is an undirected graph where each literal in the theory is associated with a node, and for each p and for every $\delta = \alpha : \beta/p$ in D, every $q \in \alpha$ and every $\sim r$ such that $r \in \beta$, there are arcs connecting all of them into one clique with p. Also, for each $p \longrightarrow q$ in W, there is an arc between p and q.

Theorem 5.1 *A network theory whose interaction graph has an induced width w^* can decide existence, membership and entailment in $O(n * 2^{w^*+1})$ when the theory is acyclic and $O(n^{w^*+2})$ when the theory is cyclic.* □

Example 5.2

Consider the set $\mathcal{P}_\Delta$ generated in example 4.6. The interaction graph is as shown in figure 4 (isolated nodes are omitted). This graph is already chordal, and if we take the ordering $\neg D, \neg L, L, M, D, W$ we see that $w^* \leq 3$, and so this network theory belongs to a class of networks for which the queries we posed can be answered in time bounded by exp(4). According to Stillman's classification [Stillman, 1990] this network theory belongs to a class whose membership problem is NP-complete. □

6 Summary and conclusions

We have presented a necessary and sufficient condition for coherence of propositional inheritance theories, provided a procedure that computes an extension, and identified tractable subsets of network default theories. The algorithm handles membership and entailment queries as well. Specifically, we have shown that network theories whose topologies have bounded induced width can be processed in polynomial time w.r.t. this parameter.

Our approach is to compile a network theory into a propositional theory such that the set of models of the latter coincides with the set of extensions of the former. Consequently, questions of coherence, membership and entailment on the network theory are equivalent to propositional satisfiability. This brings problems in non-monotonic reasoning into the familiar arenas of both propositional satisfiability and constraint satisfaction problems. Although we use here a two-step translation (from inheritance networks to default theories and then to propositional theories), it is easy to see that we can translate the inheritance network directly into a propositional theory.

Our work adds to previous research on network theories and inheritance reasoning. Etherington [Etherington, 1987] has shown a sufficient condition for coherence and presented a procedure that computes an extension of *ordered* network theories only. Stillman [Stillman, 1990] has shown that the membership problem for propositional network theories is NP-Complete and claimed to have polynomial algorithms for solving membership of a single literal in restricted subsets of network theories.

In the future we intend to extend our approach to handle preferred extensions, as formulated by Etherington and Touretzky [Touretzky, 1984], namely, to use only normal default rules, and define a partial order on the proof sequences. Using constraint network techniques, we hope to show that a most preferred extension can be obtained with the same complexity as those for finding an arbitrary one. In [Ben-Eliyahu and Dechter, 1991b] we show how this approach can be applied to any default theory.

[4] The *width* of a node in an ordered graph is the number of edges connecting it to nodes lower in the ordering. The width of an ordering is the maximum width of nodes in that ordering, and the width of a graph is the minimal width of all its orderings. The induced width is the width of the graph after it was completed to be a chordal graph.

[5] A graph is *chordal* if every cycle of length at least four has a chord.

References

[Ben-Eliyahu and Dechter, 1991a] Rachel Ben-Eliyahu and Rina Dechter. Default logic, propositional logic and constraints. In *AAAI-91: Proceedings of the 9th national conference on AI*, pages 379–385, Anaheim,CA,USA, July 1991.

[Ben-Eliyahu and Dechter, 1991b] Rachel Ben-Eliyahu and Rina Dechter. Propositional semantics for default logic. Technical Report R-172, Cognitive systems lab. UCLA, 1991.

[Dechter and Pearl, 1989] Rina Dechter and Judea Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.

[Elkan, 1990] Charles Elkan. A rational reconstruction of nonmonotonic truth maintenance systems. *Artificial Intelligence*, 43:219–234, 1990.

[Etherington, 1987] David W. Etherington. Formalizing nonmonotonic reasoning systems. *Artificial Intelligence*, 31:41–85, 1987.

[Gelfond and Lifschitz, 1990] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *The 7th international conference on Logic Programming*, pages 579–597, Jerusalem,Israel, June 1990.

[Kautz and Selman, 1991] Henry A. Kautz and Bart Selman. Hard problems for simple default logics. *Artificial Intelligence*, 49:243–279, 1991.

[Reiter and de Kleer, 1987] Raymond Reiter and Johan de Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. In *AAAI-87: Proceddings of the national conference on AI*, pages 183–188, Seattle,WA, July 1987.

[Reiter, 1980] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Stillman, 1990] Jonathan Stillman. It's not my default : The complexity of membership problems in restricted propositional default logics. In *AAAI-90: Proceedings of the 8th national conference on artificial intelligence*, pages 571–578, Boston,MA, 1990.

[Touretzky, 1984] David S. Touretzky. Implicit ordering of defaults in inheritance systems. In *AAAI-84: Proceedings of the national conference on artificial intelligence*, pages 322–325, Austin,TX, 1984.

Decision-theoretic defaults

David Poole
Department of Computer Science,
University of British Columbia,
Vancouver, B.C., Canada V6T 1Z2
email: poole@cs.ubc.ca

Abstract

This paper considers defaults as summaries of decision-theoretic deliberations. We investigate the idea that the default $e \rightarrow a$ means that a is the optimal action based on all we know (contingently) being e. It is shown how this notion of a default is nonmonotonic and has a preference for more specific defaults. It has the advantage of defaults can, in principle, be derived from lower level concepts. We thus have a rational basis for determining whether a default is correct or not. One special case considered is where the action is whether to accept some proposition as true, accept it as false or neither. This is needed to allow for conclusions to be used as premises in other defaults. It is shown that when the gain in utility of accepting a proposition depends only only on the truth of the proposition, then the acceptance of q based on evidence e depends only on whether $P(q|e)$ exceeds a threshold that is a function of the utilities for accepting q. We also give a bound on the loss (in utility terms) of using an accepted proposition in another derivation.

1 Introduction

In AI, formal default reasoning started off as a spin off from logic [Reiter, 1980; McCarthy, 1980; McDermott and Doyle, 1980]. Logic is a normative theory of correct reasoning; the hope was that by adding in a "nonmonotonic" component, a normative theory of reasoning where we jump to conclusions could be derived. Probability theory on the other hand started off as a normative theory of reasoning under uncertainty [Jeffreys, 1961], but is very quantitative in nature. Recently qualitative versions of probability theory have been proposed for default reasoning [Pearl, 1989; Neufeld, 1989; Bacchus, 1989]. One problem with all of these proposals (with some notable exceptions [Neufeld, 1989; Bacchus, 1989]) is that we cannot "take the semantics seriously"; there is no way to use the semantics to decide whether some default is correct or not. When we do take the semantics seriously it is not so obvious that the default statements say what we actually want to say.

1.1 Defaults and utilities

What one is prepared to say "yes" to depends on both utility and probabilistic information (Doyle [1989] argues this most strongly; see section 6.1).

If one is playing a game like "trivial pursuit" (where there is no penalty for saying something wrong over the penalty for saying nothing), it is better to have a wild guess at something than to say nothing. If one is in court acting as an expert witness, then one should only say what one is sure of; witnesses don't want to be caught out and have their credibility ruined. What one assents to, and so what defaults one uses, is very dependent on the situation and the utilities involved.

If one is in a closed room full of mixes of birds and someone opens up some windows high up in the room, then what one believes about the prototypical bird in the room changes as the proportion of the flying birds in the room changes[1]. At the start we may believe that the prototypical default bird in the room flies, but as the population of the birds change, after half an hour we may believe that the prototypical bird in the room does not fly. Thus probabilistic information (information about proportions of populations with certain properties) does affect the defaults we make.

In this paper we consider a formulation of defaults that takes probability and utility into consideration.

1.2 The Proposal

Other people have observed that utilities have something to do with default reasoning [Shoham, 1987; Loui, 1990; Doyle, 1989; Kadie, 1988]. In this paper we take this relationship seriously and treat defaults as decision theory summaries.

A default $e \rightarrow_A a$ means that a is the best decision out of those decisions in A if all you know is e. Note that the conclusion of a default is an action, and not a proposition as in most default frameworks (but see section 4).

The default "if you are in Vancouver in November, carry an umbrella", is of this type of a default that has an action as a conclusion and propositions as premises. This is represented as

$$in\_Vancouver \wedge is\_November \rightarrow carry\_umbrella$$

[1] This example is due to Alan Mackworth (personal communication)

The main feature of this framework are:

- We develop a meaning for defaults and inherit a calculus (albeit very weak) for reasoning with these defaults that is sound with respect to the semantics.
- We can take the semantics seriously, and argue whether or not some default is true or not. Moreover, it can be argued that these default statements are the sort of statements that correspond to everyday defaults.
- This is useful in its own right as a summary of what actions should be taken based on certain evidence. For example, in some implementations of influence diagrams (those that evaluate the diagram independently of any particular observations) [Shachter, 1986], the output is a contingency table of the output for all tuples of possible observations. One of the motivations for this paper was to allow for a more compact representation of the decisions based on different combinations of observations.
- Building on a decision theoretic base, we develop the notion of approximate reasoning, where we can have a measure on the cost of making a mistake. This is useful when we want to develop a theory of fast, but approximate reasoning.

In section 4 we consider the special case of where the actions are to accept some proposition, to accept its negation or to accept neither. This is special as it allows us to use the conclusion of the default as a premise for more inference. We analyse the possible costs of making this unsound but often reasonable rule.

2 Background

2.1 Probability

We use a standard definition of conditional Bayesian probability (e.g., [Jeffreys, 1961]), where $P(\alpha|\beta)$ is a function from two propositions into the interval $[0,1]$, where $\beta \not\equiv false$. We use the formulation based on the three axioms:

1. $P(x|x) = 1$
2. $P(\neg x|y) = 1 - P(x|y)$
3. $P(x \wedge y|z) = P(x|y \wedge z) \times P(y|z)$

The following lemma can be easily proven from the axioms and is used in this paper:

Lemma 2.1 $P(x|z) = P(x|y \wedge z) \times P(y|z) + P(x|\neg y \wedge z) \times P(\neg y|z)$

We use the symbol $\Rightarrow$ for normal logical (material) implication.

Lemma 2.2 if $y \Rightarrow z$ then $P(x|y \wedge z) = P(x|y)$.

2.2 Classical decision theory

Under classical Bayesian decision theory (e.g., [Raiffa, 1968]), we assume that there is a subjective utility function, $\mu(a,w)$ of the utility of action a if the world is w.

The expected utility of action a given evidence e, $\mathcal{E}(a,e)$ is given by

$$\mathcal{E}(a,e) = \sum_w \mu(a,w) \times P(w|e)$$

This is the utility of a averaged over all possible worlds, weighted by their probability.

3 Decision-theoretic defaults

If e is a formula in the propositional calculus, and $A = \{a_1, a_2, ...\}$ is a set of possible alternate actions (the possible actions being primitive), and $a \in A$, we write

$$e \rightarrow_A a$$

if

$$\mathcal{E}(a,e) = \max_{a_i \in A} \mathcal{E}(a_i,e).$$

In other words, $e \rightarrow_A a$ if, given all that we know (contingently — see [Poole, 1991]) is e, a is the action in A that maximises expected utility.

3.1 Nonmonotonicity

The following example shows how the meaning of defaults can allow us to derive defaults from lower level constructs. Note that, in general the user would not provide the probability and utility, but only provide the default. Because we have a formal definition of the truth of a default, we can argue about whether some default is reasonable (based on whether the underlying probabilities and defaults are reasonable). This example also shows the notion of defaults is nonmonotonic and shows how we have a preference for more specific defaults.

Example 3.1 Suppose we have the possible actions

$$A = \{say\_flies, say\_not\_flies, say\_nothing\}$$

and the following underlying utility and probability information:

$$\begin{aligned}
\mu(say_flies, flies) &= 100 \\
\mu(say_flies, \neg flies) &= -200 \\
\mu(say_not_flies, flies) &= -200 \\
\mu(say_not_flies, \neg flies) &= 100 \\
\mu(say_nothing, flies) &= 0 \\
\mu(say_nothing, \neg flies) &= 0 \\
P(flies|bird) &= 0.9 \\
P(flies|emu) &= 0.001 \\
P(bird|emu) &= 1
\end{aligned}$$

Given $bird$ we can derive the following expected utilities

$$\begin{aligned}
&\mathcal{E}(say_flies, bird) \\
&= \mu(say_flies, flies) \times P(flies|bird) \\
&\quad + \mu(say_flies, \neg flies) \times P(\neg flies|bird) \\
&= 100 \times 0.9 - 200 \times 0.1 \\
&= 70 \\
&\mathcal{E}(say_not_flies, bird) \\
&= \mu(say_not_flies, flies) \times P(flies|bird) \\
&\quad + \mu(say_not_flies, \neg flies) \times P(\neg flies|bird) \\
&= -200 \times 0.9 + 100 \times 0.1 \\
&= -170 \\
&\mathcal{E}(say_nothing, bird)
\end{aligned}$$

$$\begin{aligned} &= \mu(say\_nothing, flies) \times P(flies|bird) \\ &\quad +\mu(say\_nothing, \neg flies) \times P(\neg flies|bird) \\ &= 0 \end{aligned}$$

Thus we can derive the default

$$bird \rightarrow_A say\_flies$$

Similarly if we were given *emu*, we can compute the expected utility as:

$$\begin{aligned} \mathcal{E}(say\_flies, emu) &= -199.9 \\ \mathcal{E}(say\_not\_flies, emu) &= 99.7 \\ \mathcal{E}(say\_nothing, emu) &= 0 \end{aligned}$$

Thus we can derive the default

$$emu \rightarrow_A say\_not\_flies$$

If we are given $bird \wedge emu$ we can use lemma 2.2 to show that the same deviation works as when we are given just *emu* and so we have:

$$bird \wedge emu \rightarrow_A say\_not\_flies$$

There are two things that can be derived from this example

nonmonotonicity When we learnt new information, namely that the individual was an emu as well as a bird, we no longer derived the conclusion *say_flies*, but rather derived the conclusion *say_not_flies*. We thus change our minds when presented with different information.

specificity If we know that emus are birds, when we have both *emu* and *bird*, we make the same conclusion that we would using just *emu*. This preference for more specific defaults is true in general (section 3.2).

3.2 Specificity

One of the features of defaults that is important is the fact that more specific defaults should over-ride more general defaults. If we have $x \Rightarrow y$, then knowing x is more specific knowledge than knowing y. When we have the defaults $x \rightarrow_A a$ and $y \rightarrow_A b$, then when given $x \wedge y$ we should conclude, by specificity, a. The following proposition establishes this:

Proposition 3.2 If $x \Rightarrow y$ and $x \rightarrow_A a$ then $x \wedge y \rightarrow_A a$.

The proof of this and other propositions appears in Appendix A.

This result should not be too surprising, as the preference for more specific knowledge is common to the probabilistic formulations of defaults (see [Pearl, 1989]), as opposed to the logical formulations of defaults.

3.3 Ignoring Irrelevance

If we have some condition c such that we make the same decision whether or not c is true, then we will make that decision even if we did not know the truth of c.

Proposition 3.3 The following is valid inference:

$$\frac{e \wedge c \rightarrow_A a \quad e \wedge \neg c \rightarrow_A a}{e \rightarrow_A a}$$

Thus if would make the same decision if c is true or false, then we can ignore the value of c in our defaults. What is important here is that we can derive consequences of our defaults based on the underlying definition.

3.4 Disjunction

We cannot do arbitrary reasoning by cases. For example the disjunction rule of Pearl [1989]

$$\frac{e_1 \rightarrow_A a \quad e_2 \rightarrow_A a}{e_1 \vee e_2 \rightarrow_A a}$$

is not valid in general. It is however valid when e_1 and e_2 either imply each other or are inconsistent. Accepting this rule would lead us to Simpson's paradox [Neufeld and Horton, 1990].

3.5 Restricting the choices

Sometimes we may have fewer choices to make than at other times. The following proposition shows that if we do not eliminate the best choice, we can restrict the choices available without affecting the default.

Proposition 3.4 If $e \rightarrow_A a$ and $B \subseteq A$ such that $a \in B$ then $e \rightarrow_B a$.

We can use the following lemma to split the set of possible alternatives.

Proposition 3.5 If $e \rightarrow_A a$ and $e \rightarrow_B b$ then either $e \rightarrow_{A \cup B} a$ or $e \rightarrow_{A \cup B} b$

In the rest of the paper we assume that the set of choices of actions is fixed, and omit the subscript to $\rightarrow$.

4 Acceptance Assumption

The preceding section considered the case when the conclusion of a default was an action. Often in default reasoning, we want to use a default to conclude that some proposition is true, and then use that proposition in further reasoning.

Jon Doyle has previously propounded the idea that we expand on:

> "... we wish to use rationality as a standard for adopting assumptions by saying that an assumption should be adopted if the expected utility of holding it exceeds the expected utility of not holding it." [Doyle, 1989, p. 5]

In this section we show how to relate the action that is a conclusion of a default to the acceptance of the truth of a proposition. We consider the acceptance of a proposition as a decision like another decision. For a proposition z there are three alternate decisions that could be made:

- z^t is the decision to accept proposition z as true.
- z^f is the decision to accept z as false.
- z^u is the decision to neither accept z nor $\neg z$.

For each proposition we make the decision of whether to accept it as true, to accept it as false or to make no commitment.

Example 4.1 We write the decision to accept *flies* as true if the individual under consideration is a bird as

$$bird \rightarrow flies^t$$

This would correspond to the default in example 3.1, but the action is to accept the proposition *flies*, rather than the action to say something. Similarly the default that injured birds do not fly, can be written:

$$bird \wedge injured \rightarrow flies^f$$

This says that if all you know about some individual is that the individual is an injured bird, that it is better to assume that it does not fly, than being uncommitted about the flying ability of the individual or assuming that it does fly.

We also allow for a default to conclude that we should not assume anything about the flying ability of young birds:

$$bird \wedge young \rightarrow flies^u$$

Note that we only have a two valued logic (classical probability theory is based on every proposition being true or false in each possible world [Jeffreys, 1961]), but we have three possible actions that we can do with respect to a proposition. We assume here that it is never a good policy to assume a proposition and its negation.

Assumption 4.2 We assume the following inequalities:

$$\mu(z^f|z) < \mu(z^u|z) < \mu(z^t|z)$$

$$\mu(z^t|\neg z) < \mu(z^u|\neg z) < \mu(z^f|\neg z)$$

That is it is better to guess correctly than to be non-committal. And it is better to be non-committal than guessing wrongly[2].

The second assumption that we make is that the utilities of different propositions are in some sense independent. We can treat the gain in accepting proposition z as not affected by the truth of other propositions.

Assumption 4.3 The change in utility of accepting z or accepting $\neg z$ or accepting neither in a world w depends only on the truth of z in w.

That is, if w_1 and w_2 are two worlds that agree on the truth of z then

$$\mu(z^r, w_1) - \mu(z^s, w_1) = \mu(z^r, w_2) - \mu(z^s, w_2)$$

where $\{r, s\} \subset \{t, u, f\}$.

This assumption means that we only have to consider the gain in making the correct decision and the loss in making an incorrect decision.

Definition 4.4 If p is an atomic proposition we use the following notational schema where r and s denote different elements of $\{t, f, u\}$, and σ is a sign (one of $+$ or $\neg$) such that σp is p if σ is $+$ and σp is $\neg p$ is σ is $\neg$. We define

$${}^r\Delta^s(\sigma p) = \mu(p^s, w) - \mu(p^r, w)$$

where w is a world in which σp is true. This schema, representing 12 different formulae, denotes the change in utility made what changing our action from r to s.

[2] Analogous results to the ones below hold when the above constraints are violated — the arithmetic is slightly changed.

For example, ${}^f\Delta^t(z)$ is $\mu(z^t, w) - \mu(z^f, w)$, where z is true in w, which is the utility gained when we decide to commit to z over committing to $\neg z$ given that z is true. ${}^u\Delta^f(\neg z)$ is $\mu(z^f, w) - \mu(z^u, w)$, where z is false in w, which is the utility gained when we decide to commit to $\neg z$ over not committing to the truth of z given that z is false.

Under assumption 4.2, ${}^f\Delta^t(z)$, ${}^f\Delta^u(z)$ and ${}^u\Delta^t(z)$ are all positive. These all consist of the gain made by making a better guess given that z is true. Note also that

$${}^f\Delta^t(z) = {}^f\Delta^u(z) + {}^u\Delta^t(z).$$

Thus ${}^t\Delta^f(\neg z)$, ${}^u\Delta^f(\neg z)$ and ${}^t\Delta^u(\neg z)$ are all positive. All of the others are negative, using the equality

$${}^r\Delta^s(\sigma z) = -{}^s\Delta^r(\sigma z)$$

Lemma 4.5 For s and r each being one of t, u or f, the following holds:

$$\mathcal{E}(z^s, x) - \mathcal{E}(z^r, x) = {}^r\Delta^s(z) \times P(z|x) + {}^r\Delta^s(\neg z) \times P(\neg z|x)$$

4.1 Characterization of defaults

In this section we analyse when we can conclude a default based on the assumptions in the previous section. We first consider when one decision should be made over another decision.

Lemma 4.6

$$\mathcal{E}(z^t, x) \geq \mathcal{E}(z^f, x)$$

if and only if

$$P(z|x) \geq \frac{{}^t\Delta^f(\neg z)}{{}^f\Delta^t(z) + {}^t\Delta^f(\neg z)}$$

Lemma 4.7

$$\mathcal{E}(z^t, x) \geq \mathcal{E}(z^u, x)$$

if and only if

$$P(z|x) \geq \frac{{}^t\Delta^u(\neg z)}{{}^u\Delta^t(z) + {}^t\Delta^u(\neg z)}$$

The following theorem is a direct corollary of lemmata 4.6 and 4.7.

Theorem 4.8 $x \rightarrow z^t$ if and only if

$$P(z|x) \geq \max\left(\frac{{}^t\Delta^f(\neg z)}{{}^f\Delta^t(z) + {}^t\Delta^f(\neg z)}, \frac{{}^t\Delta^u(\neg z)}{{}^u\Delta^t(z) + {}^t\Delta^u(\neg z)}\right)$$

What is important to notice here is that the decision to accept z based on x is determined completely by a threshold on the probability $P(z|x)$ and that the threshold is a function of the utilities of the acceptance of z.

We can carry out a similar analysis of

Lemma 4.9

$$\mathcal{E}(z^u, x) \geq \mathcal{E}(z^f, x)$$

if and only if

$$P(z|x) \geq \frac{{}^u\Delta^f(\neg z)}{{}^f\Delta^u(z) + {}^u\Delta^f(\neg z)}$$

The following theorem is a direct corollary of lemmata 4.6, 4.7 and 4.9.

Theorem 4.10 $x \rightarrow z^t$ iff

$$P(z|x) \geq \max\left(\frac{{}^t\Delta^f(\neg z)}{{}^f\Delta^t(z)+{}^t\Delta^f(\neg z)}, \frac{{}^t\Delta^u(\neg z)}{{}^u\Delta^t(z)+{}^t\Delta^u(\neg z)}\right)$$

$x \rightarrow z^f$ iff

$$P(z|x) \leq \min\left(\frac{{}^t\Delta^f(\neg z)}{{}^f\Delta^t(z)+{}^t\Delta^f(\neg z)}, \frac{{}^u\Delta^f(\neg z)}{{}^f\Delta^u(z)+{}^u\Delta^f(\neg z)}\right)$$

$x \rightarrow z^u$ iff $P(z|x)$ is between these two values.

Example 4.11 Suppose we have the following utilities

$$\begin{aligned}\mu(p^u,p) = \mu(p^u,\neg p) &= 0\\ \mu(p^t,p) = \mu(p^f,\neg p) &= a\\ \mu(p^t,\neg p) = \mu(p^f,p) &= -b\end{aligned}$$

a is the prize we get for guessing right. b is the price we pay if we are wrong; both a and b are positive.

We have:

$$\begin{aligned}\frac{{}^t\Delta^f(\neg z)}{{}^f\Delta^t(z)+{}^t\Delta^f(\neg z)} &= 0.5\\ \frac{{}^t\Delta^u(\neg z)}{{}^u\Delta^t(z)+{}^t\Delta^u(\neg z)} &= \frac{b}{a+b}\\ \frac{{}^u\Delta^f(\neg z)}{{}^f\Delta^u(z)+{}^u\Delta^f(\neg z)} &= \frac{a}{a+b}\end{aligned}$$

We have the following cases of acceptance:

$a > b$

$$\begin{aligned}q \rightarrow p^t \quad &\text{if} \quad P(p|q) \geq \frac{1}{2}\\ q \rightarrow p^f \quad &\text{if} \quad P(p|q) \leq \frac{1}{2}\end{aligned}$$

Here we would never decide on p^u. We would expect to lose by being noncommittal.

$a = b$

$$\begin{aligned}q \rightarrow p^t \quad &\text{if} \quad P(p|q) \geq \frac{1}{2}\\ q \rightarrow p^u \quad &\text{if} \quad P(p|q) = \frac{1}{2}\\ q \rightarrow p^f \quad &\text{if} \quad P(p|q) \leq \frac{1}{2}\end{aligned}$$

Here, when $P(p|q) = \frac{1}{2}$, it doesn't matter which decision we make. They all have the same expected utility.

$a < b$

$$\begin{aligned}q \rightarrow p^t \quad &\text{if} \quad P(p|q) \geq \frac{b}{a+b}\\ q \rightarrow p^f \quad &\text{if} \quad P(p|q) \leq \frac{a}{a+b}\\ q \rightarrow p^u \quad &\text{if} \quad \frac{a}{a+b} \leq P(p|q) \leq \frac{b}{a+b}\end{aligned}$$

If we are very conservative we would expect that $b \gg a$. In this case we have

$$\frac{b}{a+b} \approx 1$$

The algebra of thresholding probability here is the same as the system of Bacchus [1989], but where the actual value of the threshold depends on the utilities associated with the acceptance of the conclusion of the default.

5 Approximate Reasoning

One of the features of utility-based approach to default reasoning is the ability to have a notion of the cost of getting a wrong answer. We can thus talk precisely about a tradeoff of accuracy, and consider the cost of making assumptions. Consider the following rule (called "contraction" [Pearl, 1989]):

$$\frac{\begin{array}{c}x \rightarrow y^t\\ x \wedge y \rightarrow z^t\end{array}}{x \rightarrow z^t}$$

This rule says that if we can conclude y, and use y to conclude z then we can conclude z without using y. This rule says that we can use derived conclusions as lemmata for other conclusions. This is not a valid rule of inference in the decision-theoretic defaults [Bacchus, 1989]. This is because we do not know that y is true we have only decided that we should make it true.

Pearl [1989] argues that ϵ-semantics (in which contraction is a valid rule) is an idealisation. One of the main advantages of the decision-theoretic defaults is that we can measure the cost of our idealisation. With the decision-theoretic defaults we can consider how much we can lose by applying the above rule.

Proposition 5.1 The maximum that we can lose by following the above rule is

$$(1 - th(y^t)) \times {}^t\Delta^f(\neg z)$$

where $th(y^t)$ is the threshold for accepting y, which is

$$th(y^t) = \max\left(\frac{{}^t\Delta^f(\neg z)}{{}^f\Delta^t(z)+{}^t\Delta^f(\neg z)}, \frac{{}^t\Delta^u(\neg z)}{{}^u\Delta^t(z)+{}^t\Delta^u(\neg z)}\right)$$

Example 5.2 Using the utilities of example 4.11, we find that the maximum we can lose is

$$\begin{aligned}&(1 - th(y^t)) \times {}^t\Delta^f(\neg z)\\ =\ &\frac{a}{a+b} \times (a+b)\\ =\ &a\end{aligned}$$

Even if we are extremely conservative and have a large b value, the conservatism in the acceptance of y means that we cannot lose much when we accept z.

6 Comparison with other proposals

6.1 Doyle

Doyle has also considered the role of utility and probability in default reasoning [Doyle, 1989]. This paper can be seen as following in the pioneering steps of [Doyle, 1989]

in incorporating rationality into reasoning. We go into much more detail in one case of the general framework outlined by Doyle.

Doyle [1990] motivates his rational belief revision in economic terms. However, unlike the defaults in this paper, the object level statements are not statements of preference in a utility sense. The utility is to suggest alternate definitions of belief revision. I would argue that the notion of utility of beliefs should be logically prior to the notion of rational belief revision. Once we have a notion of the utility of belief, we should be able to use this to develop a notion of rational belief revision.

Other work of Doyle [1985; 1989] has considered the problem of default reasoning as a problem of group decision making, and used the theory of group decision making for default reasoning. The group decision making and the individual decision making used in this paper are not incompatible (unless we want to claim they are the same [Doyle and Wellman, 1989]), and so these approaches should be seen as complementary to the approach propounded here.

6.2 Shoham

Shoham [1987] has argued that we should take probabilities and utilities into account when considering defaults. Here we take this suggestion seriously and consider the normative theory of decision making as a starting point. He instead develops a general framework of nonmonotonic reasoning based on ordering of interpretations. The system propounded here cannot be simply put into the framework developed by Shoham (one of the reasons is that we have automatic specificity, which one can show cannot be in any system that treats all logically equivalent formulae as equivalent [Poole, 1991]).

6.3 Loui

Loui [1990] has also proposed a mix between decision theory and defeasible reasoning. He has, however, suggested the opposite mix, namely using a form of defeasible reasoning for decision making. His motivation is very different to the motivation of this paper; it is an intriguing idea to consider whether the default system propounded here could be used as the basis for the argument system in Loui's proposal.

6.4 Bacchus

Bacchus [1989] has investigated the logic of thresholding conditional probability. All of the results of his theory can be transferred to the system in this paper. We complement Bacchus' work in that we show how straightforward decision-theoretic concerns lead us to thresholding probability.

Rather than having a constant threshold for acceptance, we have a different threshold for each proposition. While this is not inconsistent with Bacchus's results, it is interesting that we can determine exactly what the threshold should depend on. This is because we can answer the question of where the thresholds come from.

In Bacchus's system, all one can say about such rules as contraction (section 5) is that they are unsound with respect to the thresholding semantics. In the system outlined in this paper we can answer the question of how much we can lose by using these idealised rules of inference. and look at the utility of using conclusions, even if they may be mistaken.

7 Conclusion

In this paper we considered a simple idea; namely that defaults provide summaries of possible decisions that has already taken utilities and probabilities into considerations. This allows for a definition of default for which we can take the meaning seriously. I would argue that the default "birds fly" really means that if all you know about some individual is a bird, then it is good policy to assume that the individual can fly. Rather than using decision theory directly for nonmonotonic reasoning [Kadie, 1988], this paper has explored only having the summaries of good decisions as defaults.

The main result was to show that under the assumption that the utility of the choice of whether to accept a proposition depends only on the truth of the proposition (assumption 4.3), the acceptance depends on thresholds of conditional probability. Thus we get to the same system that Bacchus [1989] proposed. We have the advantage that we can derive the threshold for acceptance from utility considerations. This is one of the few proposals that can use the idea of the cost of an incorrect conclusion.

The resulting calculus is very weak. Further work can be carried out in incorporating independence assumptions, and in making assumption 4.3 more realistic. Assumption 4.3 is interesting as an idealisation, but is not practical. In practice the importance of a piece of information critically depends on what other information is true.

A Proofs

Proposition 3.2 If $x \Rightarrow y$ and $x \rightarrow a$ then $x \wedge y \rightarrow a$.

Proof: If $x \Rightarrow y$ then $P(y|x) = 1$.

$$\begin{aligned}
&\mathcal{E}(a_i, x \wedge y) \\
&= \sum_w \mu(a_i, w) \times P(w|x \wedge y) \\
&= \sum_w \mu(a_i, w) \times P(w|x) \text{ (by lemma 2.2)} \\
&= \mathcal{E}(a_i, x)
\end{aligned}$$

The result follows immediately. □

Proposition 3.3 The following is valid inference:

$$\frac{e \wedge c \rightarrow_A a \quad e \wedge \neg c \rightarrow_A a}{e \rightarrow_A a}$$

Proof: Using lemma 2.1, we have

$$\begin{aligned}
&P(w|e) \\
&= P(w|e \wedge c) \times P(c|e) \\
&\quad + P(w|e \wedge \neg c) \times P(\neg c|e) \\
&\mathcal{E}(a_i, e)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_w \mu(a_i, w) \times P(w|e) \\
&= P(c|e) \sum_w \mu(a_i, w) \times P(w|e \wedge c) \\
&\quad + P(\neg c|e) \sum_w \mu(a_i, w) \times P(w|e \wedge \neg c) \\
&= P(c|e)\mathcal{E}(a_i, e \wedge c) + P(\neg c|e)\mathcal{E}(a_i, e \wedge \neg c)
\end{aligned}
$$

We know $e \wedge c \rightarrow a$ and $e \wedge \neg c \rightarrow a$, so for each $a_i \in A$,

$$
\begin{aligned}
\text{given } \mathcal{E}(a, e \wedge c) &\geq \mathcal{E}(a_i, e \wedge c) \\
\text{and } \mathcal{E}(a, e \wedge \neg c) &\geq \mathcal{E}(a_i, e \wedge \neg c) \\
\text{then } P(c|e)\mathcal{E}(a, e \wedge c) + & \\
P(\neg c|e)\mathcal{E}(a, e \wedge \neg c) &\geq P(c|e)\mathcal{E}(a_i, e \wedge c) + \\
& \quad P(\neg c|e)\mathcal{E}(a_i, e \wedge \neg c) \\
\text{so } \mathcal{E}(a, e) &\geq \mathcal{E}(a_i, e)
\end{aligned}
$$

□

Lemma 4.5 For s and r each being one of t, u or f, the following holds:

$$\mathcal{E}(z^s, x) - \mathcal{E}(z^r, x) = {}^r\Delta^s(z) \times P(z|x) + {}^r\Delta^s(\neg z) \times P(\neg z|x)$$

Proof:

$$
\begin{aligned}
&\mathcal{E}(z^s, x) - \mathcal{E}(z^r, x) \\
&= \sum_w \mu(z^s, w) \times P(w|x) - \sum_w \mu(z^r, w) \times P(w|x) \\
&= \sum_{w:z \text{ true in } w} (\mu(z^s, w) - \mu(z^r, w)) \times P(w|x) \\
&\quad + \sum_{w:z \text{ false in } w} (\mu(z^s, w) - \mu(z^r, w)) \times P(w|x) \\
&= {}^r\Delta^s(z) \sum_{w:z \text{ true in } w} P(w|x) \\
&\quad + {}^r\Delta^s(\neg z) \sum_{w:z \text{ false in } w} P(w|x) \\
&= {}^r\Delta^s(z) \times P(z|x) + {}^r\Delta^s(z) \times P(\neg z|x)
\end{aligned}
$$

□

Lemma 4.6

$$\mathcal{E}(z^t, x) \geq \mathcal{E}(z^f, x)$$

if and only if

$$P(z|x) \geq \frac{{}^t\Delta^f(\neg z)}{{}^f\Delta^t(z) + {}^t\Delta^f(\neg z)}$$

Proof: The following sequence of inequalities are all equivalent:

$$
\begin{aligned}
\mathcal{E}(z^t, x) &\geq \mathcal{E}(z^f, x) \\
\mathcal{E}(z^t, x) - \mathcal{E}(z^f, x) &\geq 0 \\
{}^f\Delta^t(z) \times P(z|x) - {}^t\Delta^f(\neg z) \times P(\neg z|x) &\geq 0 \\
{}^f\Delta^t(z) \times P(z|x) &\geq {}^t\Delta^f(\neg z) \times (1 - P(z|x)) \\
({}^f\Delta^t(z) + {}^t\Delta^f(\neg z)) \times P(z|x) &\geq {}^t\Delta^f(\neg z) \\
P(z|x) &\geq \frac{{}^t\Delta^f(\neg z)}{{}^f\Delta^t(z) + {}^t\Delta^f(\neg z)}
\end{aligned}
$$

□

The proofs of lemmata 4.7 and 4.9 are analogous to the proof of lemma 4.6, and are omitted.

Proposition 5.1 The maximum that we can lose by following the rule of contraction is

$$(1 - th(y^t)) \times {}^t\Delta^f(\neg z)$$

where $th(y^t)$ is the threshold for accepting y, which is

$$th(y^t) = \max\left(\frac{{}^t\Delta^f(\neg z)}{{}^f\Delta^t(z) + {}^t\Delta^f(\neg z)}, \frac{{}^t\Delta^u(\neg z)}{{}^u\Delta^t(z) + {}^t\Delta^u(\neg z)}\right)$$

Proof: Suppose we have $x \rightarrow y^t$ and $x \wedge y \rightarrow z^t$. The maximum we can lose by using the rule $x \rightarrow z^t$ is given by how much we would gain by doing one of the other two actions. This is

$$max\left(\mathcal{E}(z^u, x) - \mathcal{E}(z^t, x), \mathcal{E}(z^f, x) - \mathcal{E}(z^t, x)\right)$$

For s being either of u or f, we can derive

$$
\begin{aligned}
&\mathcal{E}(z^s, x) - \mathcal{E}(z^t, x) \\
&\quad = -{}^s\Delta^t(z) \times P(z|x) + {}^t\Delta^s(\neg z) \times (1 - P(z|x)) \\
&\quad = {}^t\Delta^s(\neg z) - ({}^s\Delta^t(z) + {}^t\Delta^s(\neg z)) \times P(z|x) \\
&P(z|x) \\
&\quad = P(z|x \wedge y) \times P(y|x) + P(z|x \wedge \neg y) \times P(\neg y|x) \\
&\quad \geq P(z|x \wedge y) \times P(y|x) \\
&\quad \geq \left(\frac{{}^t\Delta^s(\neg z)}{{}^s\Delta^t(z) + {}^t\Delta^s(\neg z)}\right) \times th(y^t) \\
&\mathcal{E}(z^s, x) - \mathcal{E}(z^t, x) \\
&\quad \leq {}^t\Delta^s(\neg z) - ({}^s\Delta^t(z) + {}^t\Delta^s(\neg z)) \times \\
&\qquad \left(\frac{{}^t\Delta^s(\neg z)}{{}^s\Delta^t(z) + {}^t\Delta^s(\neg z)}\right) \times th(y^t) \\
&\quad = {}^t\Delta^s(\neg z) - {}^t\Delta^s(\neg z) \times th(y^t) \\
&\quad = {}^t\Delta^s(\neg z) \times (1 - th(y^t))
\end{aligned}
$$

So that maximum that we can lose is

$$max\left({}^t\Delta^f(\neg z) \times (1 - th(y^t)), {}^t\Delta^u(\neg z) \times (1 - th(y^t))\right)$$

which, under assumption 4.2 is ${}^t\Delta^f(\neg z) \times (1 - th(y^t))$.
□

Acknowledgements

This research was supported under NSERC grant OGPOO44121, and under Project B5 of the Institute for Robotics and Intelligent Systems.

References

[Bacchus, 1989] F. Bacchus. A modest, but semantically well founded, inheritance reasoner. In *Proc. 11th International Joint Conf. on Artificial Intelligence*, pages 1104–1109, Detroit, August 1989.

[Doyle and Wellman, 1989] J. Doyle and M. P. Wellman. Impediments to universal preference-based default theories. In R. Reiter R. J. Brachman, H. J. Levesque, editor, *Proc. First International Conf. on Principles of Knowledge Representation and Reasoning*, pages 94–102, Toronto, May 1989.

[Doyle, 1985] J. Doyle. Reasoned assumptions and pareto optimality. In *Proc. 9th International Joint Conf. on Artificial Intelligence*, pages 87–90, Los Angeles, August 1985.

[Doyle, 1989] J. Doyle. Constructive belief and rational representation. *Computational Intelligence*, 5(1):1–11, February 1989.

[Doyle, 1990] J. Doyle. Rational belief revision. In *Preprints of Third International Workshop on Nonmonotonic Reasoning*, Lake Tahoe, CA, May 1990.

[Jeffreys, 1961] H. Jeffreys. *Theory of probability.* Oxford University Press, Oxford, third edition, 1961.

[Kadie, 1988] C. M. Kadie. Rational nonmonotonic reaoning. In *Proc. Fourth Workshop on Uncertainty in Artificial Intelligence*, pages 197–204, University of Minnesota, August 1988.

[Loui, 1990] R. P. Loui. Defeasible decisions: what the proposal is and isn't. In M. Henrion et. al., editor, *Uncertainty in Artificial Intelligence 5*, pages 99–116. North Holland, 1990.

[McCarthy, 1980] J. McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1,2):27–39, 1980.

[McDermott and Doyle, 1980] D. McDermott and J. Doyle. Non-monotonic logic I. *Artificial Intelligence*, 13:41–72, 1980.

[Neufeld and Horton, 1990] E. Neufeld and J. D. Horton. Conditioning on disjunctive knowledge: Simpson's paradox in default logic. In M. Henrion et. al., editor, *Uncertainty in Artificial Intelligence 5*, pages 117–125, 1990.

[Neufeld, 1989] E. Neufeld. Defaults and probabilities; extensions and coherence. In H. J. Levesque R. J. Brachman and R. Reiter, editors, *Proc. First International Conf. on Principles of Knowledge Representation and Reasoning*, pages 312–323, Toronto, May 1989.

[Pearl, 1989] J. Pearl. Probabilistic semantics for nonmonotonic reasoning: A survey. In H. J. Levesque R. J. Brachman and R. Reiter, editors, *Proc. First International Conf. on Principles of Knowledge Representation and Reasoning*, pages 505–516, Toronto, May 1989.

[Poole, 1991] D. Poole. The effect of knowledge on belief: conditioning, specificity and the lottery paradox in default reasoning. *Artificial Intelligence*, 49:281–307, 1991.

[Raiffa, 1968] H. Raiffa. *Decision Analysis.* Addison-Wesley, 1968.

[Reiter, 1980] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1,2):81–132, 1980.

[Shachter, 1986] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34(6):871–882, November-December 1986.

[Shoham, 1987] Y. Shoham. Nonmonotonic logics: Meaning and utility. In *Proc. 10th International Joint Conf. on Artificial Intelligence*, pages 388–393, Milan, August 1987.

Hierarchical meta-logics for belief and provability: how we can do without modal logics *

Fausto Giunchiglia[1,2] and **Luciano Serafini**[1]
Mechanized Reasoning Group
[1]IRST, Povo, 38050 Trento, Italy
[2]DIST, University of Genoa, Via Opera Pia 11A, Genova, Italy
fausto@irst.it, serafini@irst.it
phone: +39 461 814440, fax: +39 461 810851

Abstract

MultiLanguage systems (ML systems) are formal systems allowing the use of multiple distinct logical languages. In this paper we introduce a class of ML systems which use a hierarchy of metatheories, each with a first order language containing names for the language below, and propose them as an alternative to modal logics.

The motivations of our proposal are technical and epistemological. From a technical point of view, we prove, among other things, that modal logics can be embedded in the corresponding ML systems. Moreover, we show that ML systems have properties not holding for modal logics and argue that these properties are justified by our intuitions. We motivate our claim by studying how they can be used in the representation of beliefs (more generally, propositional attitudes) and provability, two areas where modal logics have been extensively used.

1 Introduction and motivations

It has been argued that knowledge should be structured into sets of facts or theories (often called "contexts"); some of the many examples are [Giunchiglia and Weyhrauch, 1988; Giunchiglia, 1992a; Weyhrauch, 1980; McCarthy, 1990; Kim and Kowalski, 1990]. In [Giunchiglia, 1991; Giunchiglia and Serafini, 1991a] the authors take a further step and introduce a new general kind of formal systems allowing multiple distinct languages and call them *Multi-Language systems* (*ML systems*). In [Giunchiglia, 1991] it is argued, in fact, that providing each theory with its own language allows us to give a natural and elegant proof theoretic account of multi-contextual reasoning and, also, extra flexibility which can be exploited in the representation of many phenomena.

In this paper we focus on a particular class of ML systems which allow a hierarchy of metatheories, each using a first order language containing names for the language below and *propose them as an alternative to modal logics.* We show how modal systems can be embedded in the corresponding ML systems by providing an equivalence result between their provability relations. Moreover, we prove that the ML systems we consider have further properties, not holding in modal logic, and argue that these properties are grounded into our intuitions. To justify our claim we study how ML systems can be used in the representation of beliefs (more generally, propositional attitudes) and provability, two areas where modal logics have been extensively used; [J.Y. Halpern, 1985] and [Boolos, 1979] are some of the many references on the use of modal logics respectively on the first and the second topic.

One of our main interests is to provide foundations to the implementation of "intelligent" reasoning systems. The issue of mechanizibility and of naturalness of the interaction with the implemented system plays a central role in our research. The ideas described here have been incorporated into a system, called `GETFOL`, which gives the user the ability to define arbitrary ML systems (`GETFOL` is a total re-implementation/extension of the `FOL` system [Weyhrauch, 1980; Giunchiglia and Weyhrauch, 1991]).

The paper is structured as follows. Section 2 gives a short description of some basic notions concerning ML systems (but see [Giunchiglia, 1991] for a much longer presentation). Section 3 introduces the class $\mathcal{MR}$, the $\mathcal{MR}$ system MBK for the representation of propositional attitudes and the $\mathcal{MR}$ system MK for the representation of provability. Section 4 proves and discusses the main technical results about MBK and MK. Finally section 5 shows how $\mathcal{MR}$ systems can be extended to capture various modal logics. Proofs are skipped, for a complete treatment of the technical issues see [Giunchiglia and Serafini, 1991b].

2 ML systems

The goal of this section is to give a formalization of the idea of system with multiple languages. A detailed discussion about the definitions concerning ML systems can be found in [Giunchiglia, 1991].

An axiomatic formal system S is usually described as a triple consisting of a language L, a set of axioms $\Omega \subseteq L$ and a set of inference rules Δ, *i.e.*, $S = \langle L, \Omega, \Delta \rangle$.

*This work has been done as part of the project "MAIA", Advanced Model of Artificial Intelligence, under development at IRST.

The generalization we consider here is to take many languages and many sets of axioms while keeping one set of inference rules. We thus have the following definition of multi-language system:

Definition 2.1 (Multi-Language System) *Let I be a set of indices, $\{L_i\}_{i\in I}$, a family of languages and $\{\Omega_i\}_{i\in I}$ a family of sets of wffs such that $\Omega_i \subseteq L_i$. A* Multi-Language Formal System *(*ML system*) MS is a triple $\langle\{L_i\}_{i\in I}, \{\Omega_i\}_{i\in I}, \Delta\rangle$ where $\{L_i\}_{i\in I}$ is the* Family of Languages, *$\{\Omega_i\}_{i\in I}$ is the* Family of sets of Axioms *and Δ is the* Deductive machinery *of MS.*

If A is a wff of a language L, we say that A is an L-wff. Each language L_i is associated with its theory (defined as the set of L_i-wffs which can be proved by applying the deduction machinery to the axioms). *What can be derived is bound by language*: certain formulas (like the conjunction of two theorems in two distinct theories) cannot be derived simply because there is no language in which they can be expressed.

Even if definition 2.1 is more general, allowing arbitrary formal languages and deductive machinery, in this paper (and in most of the work done so far), we concentrate on first order languages and adopt a suitable modification of the natural deduction formalism, notation and terminology defined in [Prawitz, 1965]. Moreover, as we want to make effective use of the multiple languages, we define the inference rules in a way to take into account the language the wffs are extracted from. We write $\langle A, i\rangle$ to mean A and that A is a L_i-wff. The deduction machinery Δ, is therefore defined as a set of inference rules, written as:

$$\frac{\langle A_1, i_1\rangle \quad \ldots \quad \langle A_n, i_n\rangle}{\langle A, i\rangle}\ \iota \tag{1}$$

or as:

$$\frac{\begin{array}{cc}[\langle B_1, j_1\rangle] & [\langle B_m, j_m\rangle] \\ \langle A_1, i_1\rangle \ldots \langle A_n, i_n\rangle \quad \langle A_{n+1}, i_{n+1}\rangle \ldots \langle A_{n+m}, i_{n+m}\rangle & \end{array}}{\langle A, i\rangle}\ \delta \tag{2}$$

Picture (2) represents a rule δ discharging the assumptions $\langle B_1, j_1\rangle, \ldots, \langle B_m, j_m\rangle$ [Prawitz, 1965]. Notice that in general, inference rules have premises and conclusions belonging to different languages. The rules whose premises and conclusions belong to the same language L_i are called *L_i-rules*, the others *bridge rules* [Giunchiglia, 1991]. L_i-rules allow to draw consequences inside a theory while bridge rules allow to export results from one theory to another. One example of L_i-rule for a theory i is modus ponens

$$\frac{\langle A \rightarrow B, i\rangle \quad \langle A, i\rangle}{\langle B, i\rangle}$$

one example of bridge rule between the theories i, j and the theory k is *multicontextual modus ponens* [Giunchiglia, 1992a]

$$\frac{\langle A \rightarrow B, i\rangle \quad \langle A, j\rangle}{\langle B, k\rangle}$$

The meaning of the two rules is very different. The first allows us to derive B inside the theory i just because we have derived $A \rightarrow B$ and A in the same theory. Thus, for instance, if we take the theory i to represent the beliefs of an agent a_i, this means providing the agent a_i with the ability of using modus ponens. Multicontextual modus ponens allows us to derive B in the theory k just because we have derived $A \rightarrow B$ in the theory i and A in the theory j. If we take the theories i, j, k to represent the beliefs of the agents a_i, a_j, a_k, this means that a_k will be able to take B as known, not because he has derived it, but because this is the result of the interaction of the results derived by a_i and a_j. The assertion of B in k is not the result of a deduction in k but, rather the result of the "propagation" of reasoning from i, j into k. ([Giunchiglia, 1992a] describes these issues in more detail.)

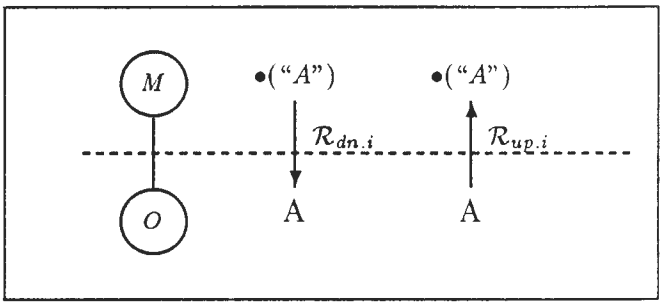


Figure 1: The family $\mathcal{MR}$

Deductions are trees of wffs built starting from a finite number of assumptions and axioms, possibly belonging to distinct languages, and applying a finite number of inference rules. *Any deduction can be seen as composed of subdeductions in distinct languages, obtained by repeated applications of L_i-rules, any two or more subdeductions being concatenated by one or more applications of bridge rules.*

3 The class $\mathcal{MR}$

Informally, $\mathcal{MR}$ systems have the following properties:

(i) the languages are ordered in a hierarchy;

(ii) each language in the hierarchy has names for the wffs of the level below;

(iii) any two adjacent languages in the hierarchy, say M and O, are linked only by two bridge rules which are variations of the multi-language version of reflection up and reflection down (as described in [Giunchiglia and Smaill, 1989]). In other words the bridge rules are of the form:

$$\frac{\langle A, O\rangle}{\langle \bullet(\text{"}A\text{"}), M\rangle}\ \mathcal{R}_{up.} \qquad \frac{\langle \bullet(\text{"}A\text{"}), M\rangle}{\langle A, O\rangle}\ \mathcal{R}_{dn.}$$

where "•" is a unary predicate.

Figure 1 gives a graphical representation of the basic structure of the elements of $\mathcal{MR}$.

Each $\mathcal{MR}$ system is a hierarchical meta-logic, in the sense that each pair of connected theories, O and M,

satisfies the following conditions:

$$\vdash_O A \textit{ if and only if } \vdash_M \bullet(\text{“}A\text{”}) \quad (3)$$

$$\vdash_M \bullet(\text{“}A \rightarrow B\text{”}) \rightarrow (\bullet(\text{“}A\text{”}) \rightarrow \bullet(\text{“}B\text{”})) \quad (4)$$

which are the weakest conditions that guarantee the object/meta relation between two theories.

In this section we present and discuss two important instances of $\mathcal{MR}$ systems, that is MBK for the representation of propositional attitudes and MK for the representation of metatheoretic theorem proving.

3.1 MBK: reasoning about propositional attitudes

MBK is the basic system for the representation of propositional attitudes. To keep things simple we consider the single agent case. The generalization to the multi-agent case is straightforward.

The idea underlying the formalization of propositional attitudes is that there is an agent, let us call him a, who is acting in a world, and has both beliefs about this world, and beliefs about his own beliefs. For a proposition A about the state of the world, $Bl(\text{“}A\text{”})$ means that A itself is believed by a or, in other words, that A holds in a's view of the world; similarly $Bl(\text{“}Bl(\text{“}A\text{”})\text{”})$ means that $Bl(\text{“}A\text{”})$ is believed by a, *i.e.* A holds in a's view of his beliefs of the world, and so on. In other words, a "sits on top" of his beliefs and is able to reason on the reification of his belief A, that is $Bl(\text{“}A\text{”})$, in a sort of "metaview" of a believing A.

This process of reification can be iterated through a chain a metaviews; a's beliefs are thus the facts derived in a top theory which has a metaview, a metametaview, a meta...metaview of a's own beliefs.

To formalize the notion of belief in a multi-language framework we have a chain of theories, called *views*, where the theory above "sees" the theory below via reflection principles which allow the derivation of $Bl(\text{“}A\text{”})$ from A and viceversa (see figure 2). This chain has a top theory which is a's beliefs, that is, its basic beliefs about the world and his view of all the possible nestings of the belief predicate. In the case of the ideal reasoner, which we consider in the following, we need to have an infinitely descending chain and thus no bottom theory (one more level of nesting of the belief predicate corresponds to one more theory in the chain, see figure 2).

As each theory is "above" an infinite chain and each level corresponds to a level of nesting of the belief predicate, all the languages in MBK must have the same expressibility, *i.e.* they must have the same notion of wellformedness.

To define MBK we need a way to index theories in the infinitely descending chain. We index the top theory with 0, the one below with 1 and so on.

In MBK the languages $L(Bl)$, are thus obtained from a propositional language L as follows:

$$L_0 = L$$

$$L_{n+1} = L_n \cup \{Bl(\text{“}A\text{”}) : A \text{ is an } L_i\text{-wff}\}$$

$$L(Bl) = \bigcup_{n \in \omega} L_n$$

Here L is the language used to express the basic facts about the world.

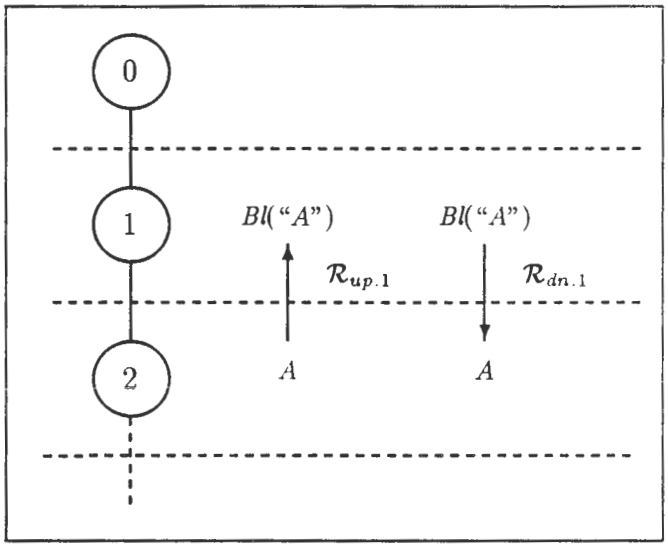


Figure 2: The system MBK

Definition 3.1 (MBK) *Let L be a propositional language. Then $MBK = \langle \{L_i\}_{i \in \omega}, \{\Omega_i\}_{i \in \omega}, \Delta \rangle$, is such that, for every $i \in \omega$, $L_i = L(Bl)$, $\Omega_i = \emptyset$ and Δ contains the following rules:*

$$\frac{\begin{matrix}[\langle A, i\rangle] \\ \langle B, i\rangle\end{matrix}}{\langle A \rightarrow B, i\rangle} \rightarrow I_i \qquad \frac{\langle A, i\rangle \quad \langle A \rightarrow B, i\rangle}{\langle B, i\rangle} \rightarrow E_i$$

$$\frac{\begin{matrix}[\langle \neg A, i\rangle] \\ \langle \bot, i\rangle\end{matrix}}{\langle A, i\rangle} \bot_i$$

$$\frac{\langle A, i+1\rangle}{\langle Bl(\text{“}A\text{”}), i\rangle} \mathcal{R}_{up.i} \qquad \frac{\langle Bl(\text{“}A\text{”}), i\rangle}{\langle A, i+1\rangle} \mathcal{R}_{dn.i}$$

Restrictions: the $\mathcal{R}_{up.i}$-rule can be applied only if the index of every undischarged assumption $\langle A, i\rangle$ depends on, is lower than or equal to i. $\bot_i$ can be applied if A is not of the form $\bot$.

The idea underlying the definition of MBK is that the only bridge rules are reflection up and down between any two adjacent theories and that any theory i has a set of inference rules which is complete for classical propositional logic (this ensures that the i-th theory contains all the tautologies).

A lot of people in the AI and cognitive science community have argued in favour of this kind of "distributed" representation of beliefs and deductions. (Notice that in the multiagent case, instead of a hierarchy, we have a tree of theories. In this tree, each theory is above n theories, one for each agent; each of these n theories represents the beliefs of an agent as "seen" by the theory above them). For instance Wilks [Wilks and Biem, 1979], in his work on belief ascription, speech acts and so on, advocates the use of distinct sets of beliefs; Fauconnier [Fauconnier, 1985] has a mental space theory which uses environment-like entities; Dinsmore in [Dinsmore, 1991] formalizes

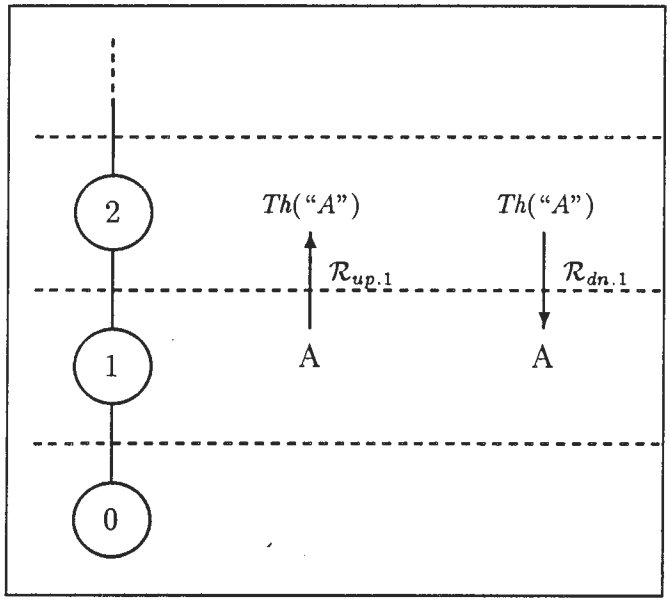


Figure 3: The system MK

agents' beliefs by belief spaces, each of which can contain further belief spaces for the formalization of the agents' nested beliefs. Konolige formalizes agents' nested beliefs using introspective machines [Konolige, 1984; Konolige, 1985], which are very similar to theories. Kim and Kowalski in [Kim and Kowalski, 1990] propose a very similar approach in the area of logic programming.

Notice that Perlis [Perlis, 1985] argues explicitly against the use of multiple hierarchical theories, the main argument being one of quantification. He argues, rightly, that we do not want to quantify over all the different levels. His point does not apply here as we are dealing with the propositional case; on the other hand, we argue that, even in the first order case, his observation does not rule out the use of multiple levels and, more generally, of multiple contexts. Even in the first order case we do not need to quantify over theories unless this is exactly the kind of reasoning we want to do. The basic intuition, underlying *all* our work, is that reasoning is always contextual and that, therefore, all the arguments are implicitly bounded to the current context and to the contexts it implicitly refers to (see also [Giunchiglia, 1992a; McCarthy, 1991]). We never quantify explicitly over contexts unless we want to make the contextuality of reasoning explicit.

3.2 MK: reasoning with metatheories

In metatheoretic reasoning, one usually starts with the object theory and then defines its metatheory, its metametatheory and so on. Analogously, in MK, the bottom theory is any object theory, the theory at level 1 is its metatheory, the theory at level 2 is its metametatheory and so on. Thus, if L is the propositional language of the object theory, for any natural number i ($i \in \omega$), L_i is inductively defined as follows:

$$L_0 = L$$

$$L_{i+1} = L_i \cup \{Th(\text{“}A\text{”}) : A \text{ is an } L_i\text{-wff}\}$$

where: Th is an unary predicate and “A” is an individual constant (which acts as the *name* of A).

Definition 3.2 (MK) *Let L be a propositional language. Then $MK = \langle \{L_i\}_{i\in\omega}, \{\Omega\}_{i\in\omega}, \Delta\rangle$, is such that, for every $i \in \omega$, $\Omega_i = \emptyset$ and Δ contains the following rules:*

$$\frac{\begin{array}{c}[\langle A, i\rangle] \\ \langle B, i\rangle\end{array}}{\langle A \rightarrow B, i\rangle}\ \rightarrow I_i \qquad \frac{\langle A, i\rangle \quad \langle A \rightarrow B, i\rangle}{\langle B, i\rangle}\ \rightarrow E_i$$

$$\frac{\begin{array}{c}[\langle \neg A, i\rangle] \\ \langle \bot, i\rangle\end{array}}{\langle A, i\rangle}\ \bot_i$$

$$\frac{\langle A, i\rangle}{\langle Th(\text{“}A\text{”}), i+1\rangle}\ \mathcal{R}_{up.i} \qquad \frac{\langle Th(\text{“}A\text{”}), i+1\rangle}{\langle A, i\rangle}\ \mathcal{R}_{dn.i}$$

Restrictions: the $\mathcal{R}_{up.i}$-rule can be applied only if the index of every undischarged assumption $\langle A, i\rangle$ depends on, is strictly greater than i. $\bot_i$ can be applied if A is not of the form $\bot$.

Figure 3 gives a graphical representation of the structure of MK.

Analogously to MBK, in MK reflection up and down between any two adjacent theories are the only two bridge rules and any theory i can prove all the tautologies. Most of the intuitions underlying MK are reported in [Giunchiglia and Serafini, 1991b].

3.3 Some notes about MK and MBK

MBK and MK are similar but different. Not least, they use different hierarchies of languages. The differences between MBK and MK can be best summarized by pointing out that in MK we have only one object theory and an infinite number of metatheories while in MBK we have one meta...metatheory and an infinite number of object theories. These differences, which are substantial and seem motivated by our intuitions, do not seem formalizable in modal logics. Our explanation of this fact is that, in modal logics, collapsing all the languages in one, one loses track of how, depending on the application, language is used in the hierarchy of nestings of the proof/belief predicate. Note that the union of all the languages in MK is the same as the union of all the languages in MBK modulo the substitution of Th with Bl. Having a unique language has caused a certain degree of confusion between provability and belief; for instance in [Konolige, 1984] belief has been modeled as provability.

Some basic theorems in MBK and proofs are listed below. The proofs are examples of how deductions are constructed in MBK.

Proposition 3.1 *For any wff A and B and any $i \in \omega$, (i)-(iii) are theorems of MBK.*

(i) $\langle Bl(\text{“}A \rightarrow B\text{”}) \rightarrow (Bl(\text{“}A\text{”}) \rightarrow Bl(\text{“}B\text{”})), i\rangle$

(ii) $\langle Bl(\text{“}\bot\text{”}) \rightarrow Bl(\text{“}A\text{”}), i\rangle$

(iii) $\langle \neg Bl(\text{“}\bot\text{”}) \rightarrow (Bl(\text{“}A\text{”}) \rightarrow \neg Bl(\text{“}\neg A\text{”})),\ i\rangle$

Proof

(i)

$$\dfrac{\dfrac{\dfrac{\dfrac{\langle Bl(\text{“}A \rightarrow B\text{”}),\ i\rangle}{\langle A \rightarrow B,\ i+1\rangle}\,\mathcal{R}_{dn.i} \quad \dfrac{\langle Bl(\text{“}A\text{”}),\ i\rangle}{\langle A,\ i+1\rangle}\,\mathcal{R}_{dn.i}}{\langle B,\ i+1\rangle}\,{\rightarrow}\mathrm{E}_{i+1}}{\langle Bl(\text{“}B\text{”}),\ i\rangle}\,\mathcal{R}_{up.i}}{\langle Bl(\text{“}A\text{”}) \rightarrow Bl(\text{“}B\text{”}),\ i\rangle}\,{\rightarrow}\mathrm{I}_i}{\langle Bl(\text{“}A \rightarrow B\text{”}) \rightarrow (Bl(\text{“}A\text{”}) \rightarrow Bl(\text{“}B\text{”})),\ i\rangle}\,{\rightarrow}\mathrm{I}_i$$

(ii)

$$\dfrac{\dfrac{\dfrac{\dfrac{\langle Bl(\text{“}\bot\text{”}),\ i\rangle}{\langle \bot,\ i+1\rangle}\,\mathcal{R}_{dn.i}}{\langle A,\ i+1\rangle}\,\bot_{i+1}}{\langle Bl(\text{“}A\text{”}),\ i\rangle}\,\mathcal{R}_{up.i}}{\langle Bl(\text{“}\bot\text{”}) \rightarrow Bl(\text{“}A\text{”}),\ i\rangle}\,{\rightarrow}\mathrm{I}_i$$

(iii)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\langle Bl(\text{“}A\text{”}),\ i\rangle}{\langle A,\ i+1\rangle}\,\mathcal{R}_{dn.i} \quad \dfrac{\langle Bl(\text{“}\neg A\text{”}),\ i\rangle}{\langle \neg A,\ i+1\rangle}\,\mathcal{R}_{dn.i}}{\langle \bot,\ i+1\rangle}\,{\rightarrow}\mathrm{E}_{i+1} \quad \dfrac{}{\langle Bl(\text{“}\bot\text{”}),\ i\rangle}\,\mathcal{R}_{up.i}}{\langle \neg Bl(\text{“}\bot\text{”}),\ i\rangle}\,{\rightarrow}\mathrm{E}_i}{\langle \bot,\ i\rangle}\,\bot_i}{\langle Bl(\text{“}A\text{”}) \rightarrow \neg Bl(\text{“}\neg A\text{”}),\ i\rangle}\,{\rightarrow}\mathrm{I}_i}{\langle \neg Bl(\text{“}\bot\text{”}) \rightarrow (Bl(\text{“}A\text{”}) \rightarrow \neg Bl(\text{“}\neg A\text{”})),\ i\rangle}\,{\rightarrow}\mathrm{I}_i$$

Q.E.D.

Under the obvious interpretation (read $Bl(\text{“}A\text{”})$ as $\Box A$) the three theorems above hold also for modal K. In particular the translation of the first theorem is the axiom characterizing modal K (that is $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$).

The wffs obtained by replacing Bl with Th in (i), (ii) and (iii) of proposition 3.1 are theorems of MK as well. Their proofs can be obtained by replacing i with $i+2$ (and leaving $i+1$ as it occurs) in the proof of the corresponding MBK theorems.

4 The main results about MBK

This and the following section describe the main technical results for MBK. Unless explicitly stated the contrary, all the theorems hold also for MK, modulo some obvious syntactic modifications.

Let L be a propositional language. We define the modal language $L(\Box)$ as the minimal set of wffs built with the usual rules for the logical connectives plus the following rule: if A is a $L(\Box)$-wff then $\Box A$ is a $L(\Box)$-wff. The modal system K is characterized by the axiom schema $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$ and by the *necessitation rule*: *if* $\vdash_K A$ *then* $\vdash_K \Box A$. To carry out the proofs of the theorems listed in these pages, we have used Bull and Segerberg's version of modal K, as described in [Bull and Segerberg, 1984].

The mapping $(.)^*$ from $L(\Box)$-wffs to $L(Bl)$-wffs is then defined as follows:

(i) If A is a propositional constant then $A^* = A$;

(ii) $(.)^*$ distributes over the propositional connectives;

(iii) $(\Box A)^* = Bl(\text{“}A^*\text{”})$.

$(.)^*$ is an isomorphism with inverse, which we write $(.)^+$.

The first result is a form of syntactic equivalence between provability in K and in MK.

Theorem 4.1 *If A is an $L(Bl)$-wff then for any $i \in \omega$, $\vdash_{MBK} \langle A,\ i\rangle$, if and only if $\vdash_K A^+$.*

Corollary 4.1 (Consistency) *For every $i \in \omega$, $\langle \bot,\ i\rangle$ is not a theorem in MBK.*

Notice that in [Rivieres and Levesque, 1986], Des Rivieres and Levesque show how Montague and Thomason's negative results [Montague, 1974; Thomason, 1980] can be avoided with a careful translation of modal logics into first order logics. Besides the differences in the motivations, the main technical difference between the results stated by theorem 4.1 and corollary 4.1 and those described in [Rivieres and Levesque, 1986] is that they keep a unique language while we propose a hierarchy of distinct, possibly different (as in MK) languages. [Attardi and Simi, 1991] describes another approach where reflection rules are used still keeping consistency; the main technical difference with our work is, again, that they use a unique (amalgamated) language.

Theorem 4.1 does not mean that K and MBK's provability and, more important, derivability relations can be put into a one-to-one mapping. There are further properties which hold specifically for MBK and MK but not for K. Some of these properties, described in the following, involve the multilanguage structure of MBK and concern the propagation of inconsistency through the hierarchy of theories.

Theorem 4.2 *If $\Gamma \vdash_{MBK} \langle \bot,\ i\rangle$ then $\Gamma \vdash_{MBK} \langle \bot,\ j\rangle$ for any $j \geq i$.*

Equivalently: if $\Gamma \nvdash_{MBK} \langle \bot,\ i\rangle$ then $\Gamma \nvdash_{MBK} \langle \bot,\ j\rangle$ for any $j \leq i$.

Note that the converse of theorem 4.2 does not hold; deriving $\bot$ in one theory does not yield the derivation of $\bot$ in the theories above it. Because of the locality of deductions (inside a theory) and the filtering performed by reflection up, it is impossible to propagate the inconsistency upwards. *Local inconsistency* (inconsistency inside a theory) *does not imply global inconsistency* (inconsistency everywhere, in our case in all the theories), as it *does* happen in human reasoning and *does not* happen in the "usual" logical systems.

This property is highlighted by the following result.

Theorem 4.3 $\langle \bot,\ i+1\rangle \nvdash_{MBK} \langle \bot,\ i\rangle$.

As far as propositional attitudes are concerned, the localization of reasoning in general and that of inconsistency in particular has often been argued to be a property of common sense reasoning (see for instance [Fagin and Halpern, 1988]). The intuition behind this result is that we can have a believer which reasons consistently about inconsistent beliefs and, more generally, that it is possible to have consistent beliefs about inconsistent beliefs. Analogously, in metatheoretic theorem proving, a consistent metatheory to an inconsistent theory can be used to reason (inside the system, not only in its informal metatheory or in the code implementing it) consistently

about inconsistency. One goal could, for instance, to identify the set of assumptions/ axioms generating the inconsistency.

A stronger result can be stated for MK.

Theorem 4.4 *For every finite set of wffs* Γ, *there exists a wff* $\langle A, i\rangle$ *such that* $\Gamma \nvdash_{MK} \langle A, i\rangle$.

Theorem 4.4 says that, as long as we add finitely many (possibly contradictory) assumptions to it, MK cannot get into inconsistency. This result can actually be generalized to axioms. Thus, as long as we allow only finite sets of axioms, there will always be a way to "get out" of an inconsistent situation simply by going high enough in the hierarchy of theories. The property described by theorem 4.4 does not hold in MBK, as in MBK there exists a top theory. This, agrees with the intuition that an inconsistent believer, by definition, cannot reason consistently about his own beliefs.

5 ML systems for the other modal logics

In modal logic one usually extends K to obtain stronger logics, for instance T, K4, S5, by adding appropriate axioms. With ML systems, an analogous result can be obtained by adding suitable bridge rules. For instance ML systems which are equivalent to T, K4, K45, S4 and S5 can be elegantly obtained by adding bridge rules to MBK or to MK.

Definition 5.1 *For any* $i \geq 0$, *let* $\mathcal{T}_i$, $\mathcal{S}4_i$ *and* $\mathcal{S}5_i$ *be the following bridge rules:*

$$\frac{\langle A, i+1\rangle}{\langle A, i\rangle}\ \mathcal{T}_i$$

$$\frac{\langle Th(\text{“}A\text{”}), i\rangle}{\langle Th(\text{“}A\text{”}), i+1\rangle}\ \mathcal{S}4_i$$

$$\frac{\langle \neg Th(\text{“}A\text{”}), i\rangle}{\langle \neg Th(\text{“}A\text{”}), i+1\rangle}\ \mathcal{S}5_i$$

Restrictions: $\mathcal{T}_i$ *has the same restriction as* $\mathcal{R}_{up.i}$.

Then MBT, MBK4, MBK45, MBS4 *and* MBS5 *are the ML systems obtained from MBK by adding, for any* i *the bridge rules* $\mathcal{T}_i$, $\mathcal{S}4_i$, $\mathcal{S}4_i$ *plus* $\mathcal{S}5_i$, $\mathcal{T}_i$ *plus* $\mathcal{S}4_i$ *and* $\mathcal{T}_i$ *plus* $\mathcal{S}5_i$, *respectively.*

Let us write MBX to mean one among MBT, MBK4, MBK45, MBS4 and MBS5 and X to mean the corresponding modal system (T, K4, K45, S4 and S5 respectively). We can thus prove the following theorem.

Theorem 5.1 *For any wff* A *and any* $i \in \omega$, $\vdash_{MBX} \langle A, i\rangle$, *if and only if* $\vdash_X A^+$.

Table 1 shows how the translation of the axioms of a modal system X are theorems in the corresponding ML system MBX.

MBX systems give a first positive feedback on the conjecture, discussed in [Giunchiglia, 1991; Giunchiglia, 1992a; Giunchiglia and Serafini, 1991a], that we can concentrate a lot of the "interesting research" on ML systems on the search for "suitable" bridge rules. This corresponds to the intuition that *many reasoning phenomena can be modelled as contextual reasoning by controlling the propagation of consequences among theories.*

6 Conclusion

The main goal of this paper has been to propose multilanguage systems which can be used, at least in some problem domains, in place of modal logics. This project has been carried out from two perspectives.

From a technical point of view, we have given various equivalence results with the most common modal logics. From a representational point of view, we have shown that multilanguage systems have properties not holding in modal logics and have argued that these properties are motivated by our intuitions. For instance we have introduced two ML systems, MBK and MK, both equivalent to modal K but with different characteristics. In particular MK is constructed by starting from an object theory and by progressively adding metatheories in an infinitely ascending chain; this corresponds to the intuition that we build a metatheory starting from the object theory it describes. MBK, on the other hand, is constructed by starting from the top meta...metatheory and by progressively adding an infinitely descending chain of object theories; this corresponds to the intution that a believer has a meta...metaview of the world and of his beliefs. Each new view is a new object theory for the believer to reason about. Moreover, we have shown that ML systems have other interesting properties; for instance local inconsistency (inconsistency of a theory) does not imply global inconsistency (inconsistency of all the theories) and a finite set of axioms cannot make MK globally inconsistent.

Notice that it is our deeper belief that multilanguage systems can be used to provide a unifying and foundational framework for the representation of knowledge and common sense reasoning. The idea is that reasoning is mainly contextual and that multilanguage systems provide a framework for the formalization of context-based reasoning. The work on modal logics is only one of the many examples which are being studied inside the Mechanized Reasoning Group; some other examples are: abstract reasoning, metatheoretic reasoning, reasoning by analogy, reasoning about time, reasoning about distinct subject matters.

Acknowledgments

The work with Alan Smaill, Alex Simpson, Carolyn Talcott, Paolo Traverso and Richard Weyhrauch has provided some of the intuitions and motivations underlying the work described in this paper. The discussions with Enrico Giunchiglia and John McCarthy have provided useful insights into the notion of context. David Israel, Kurt Konolige and Vladimir Lifschitz have provided useful feedback. Alex Simpson has proof checked some versions of this paper and corrected some of the mistakes.

translation of X axioms	MBX
$\langle Bl(\text{“}A\text{”}) \rightarrow A,\ i\rangle$	$\dfrac{\dfrac{\dfrac{\langle Bl(\text{“}A\text{”}),\ i\rangle}{\langle A,\ i+1\rangle}\ \mathcal{R}_{dn.i}}{\langle A,\ i\rangle}\ \mathcal{T}_i}{\langle Bl(\text{“}A\text{”}) \rightarrow A,\ i\rangle}\ \rightarrow\!\mathrm{I}_i$
$\langle Bl(\text{“}A\text{”}) \rightarrow Bl(\text{“}Bl(\text{“}A\text{”})\text{”}),\ i\rangle$	$\dfrac{\dfrac{\dfrac{\langle Bl(\text{“}A\text{”}),\ i\rangle}{\langle Bl(\text{“}A\text{”}),\ i+1\rangle}\ \mathcal{S}4_i}{\langle Bl(\text{“}Bl(\text{“}A\text{”})\text{”}),\ i\rangle}\ \mathcal{R}_{up.i}}{\langle Bl(\text{“}A\text{”}) \rightarrow Bl(\text{“}Bl(\text{“}A\text{”})\text{”}),\ i\rangle}\ \rightarrow\!\mathrm{I}_i$
$\langle \neg Bl(\text{“}A\text{”}) \rightarrow Bl(\text{“}\neg Bl(\text{“}A\text{”})\text{”}),\ i\rangle$	$\dfrac{\dfrac{\dfrac{\langle \neg Bl(\text{“}A\text{”}),\ i\rangle}{\langle \neg Bl(\text{“}A\text{”}),\ i+1\rangle}\ \mathcal{S}4_i}{\langle Bl(\text{“}\neg Bl(\text{“}A\text{”})\text{”}),\ i\rangle}\ \mathcal{R}_{up.i}}{\langle \neg Bl(\text{“}A\text{”}) \rightarrow Bl(\text{“}\neg Bl(\text{“}A\text{”})\text{”}),\ i\rangle}\ \rightarrow\!\mathrm{I}_i$

Table 1: Proofs of the translation of the modal axioms

References

[Attardi and Simi, 1991] G. Attardi and M. Simi. Reflections about reflection. In *Principles of Knowledge Representation and Reasoning, proceedings of the second international conference*, pages 22–31. Morgan Kaufmann, 1991.

[Boolos, 1979] G. Boolos. *The Unprovability of Consistency, an essay in modal logic.* Cambridge University Press, 1979.

[Bull and Segerberg, 1984] R. Bull and K. Segerberg. Basic Modal Logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume III, pages 1–88. D. Reidel Publishing Company, 1984.

[Dinsmore, 1991] J. Dinsmore. *Partitioned Representations.* Kluwer Academic Publisher, 1991.

[Fagin and Halpern, 1988] R. Fagin and J.Y. Halpern. Belief, awareness, and limited reasoning. *Journal of Artificial Intelligence*, 34:39–76, 1988.

[Fauconnier, 1985] G. Fauconnier. *Mental Spaces: aspects of meaning construction in natural language.* MIT Press, 1985.

[Giunchiglia and Serafini, 1991a] F. Giunchiglia and L. Serafini. Multilanguage first order theories of propositional attitudes. In *Proceedings 3rd Scandinavian Conference on Artificial Intelligence*, Roskilde University, Denmark, 1991. Also available as IRST-Technical Report no. 9001-02.

[Giunchiglia and Serafini, 1991b] F. Giunchiglia and L. Serafini. Multilanguage hierarchical logics (or: how we can do without modal logics). Technical Report 9110-07, IRST, Trento, Italy, 1991. Submitted to Artificial Intelligence.

[Giunchiglia and Smaill, 1989] F. Giunchiglia and A. Smaill. Reflection in constructive and non-constructive automated reasoning. In J. Lloyd, editor, *Proc. Workshop on Meta-Programming in Logic Programming.* MIT Press, 1989. IRST Technical Report 8902-04. Also available as DAI Research Paper 375, Dept. of Artificial Intelligence, Edinburgh.

[Giunchiglia and Weyhrauch, 1988] F. Giunchiglia and R.W. Weyhrauch. A multi-context monotonic axiomatization of inessential non-monotonicity. In D. Nardi and P. Maes, editors, *Meta-level architectures and Reflection.* North Holland, 1988. Also available as Technical Report 9105-02, MRG-DIST, University of Genova, Italy.

[Giunchiglia and Weyhrauch, 1991] F. Giunchiglia and R.W. Weyhrauch. FOL User Manual - FOL version 2. Technical Report 9107-02, DIST - University of Genova, Genova, Italy, 1991.

[Giunchiglia, 1991] F. Giunchiglia. Multilanguage systems. In *Proceedings of AAAI 1991 Spring Symposium on Logical Formalizations of Commonsense Reasoning*, 1991. IRST-Technical Report no. 9011-17.

[Giunchiglia, 1992a] F. Giunchiglia. Contextual reasoning. To appear in Epistemologia, 1992.

[J.Y. Halpern, 1985] Y. Moses J.Y. Halpern. A guide to the modal logics of knowledge and belief: preliminary draft. In *Proceedings IJCAI-85, Los Angeles, CA.* Morgan Kaufmann Publ. Inc., 1985.

[Kim and Kowalski, 1990] J.S. Kim and R.A. Kowalski. An application of amalgamated logic to multi-agent belief. In *Proceedings of the Second Workshop on Meta-programming in Logic, META90*, pages 272–283. K.U. Leuven, 1990.

[Konolige, 1984] K. Konolige. A deduction model of belief and its logics. Technical note 326, SRI, Artificial Intelligence Center, 1984. Stanford Computer Science Dept. PhD. thesis.

[Konolige, 1985] K. Konolige. A computational theory of belief introspection. In *Proc. 9th IJCAI conference*, 1985.

[McCarthy, 1990] J. McCarthy. Generality in Artificial Intelligence. In J. McCarthy, editor, *Formalizing Common Sense - Papers by John McCarthy*, pages 226–236. Ablex Publishing Corporation, 1990.

[McCarthy, 1991] J. McCarthy. Notes on formalizing context. Unpublished, 1991.

[Montague, 1974] R. Montague. Syntactical treatment of modality, with corollaries on reflections principles and finite axiomatizability. In *Formal Philosophy, selected papers of Richard Montague*, pages 286–302, 1974. Originally published in 'Acta Philosophica Fennica' 16:153-167 (1963).

[Perlis, 1985] D. Perlis. Languages with Self-Reference I: Foundations. *Artificial Intelligence*, 25:301–322, 1985.

[Prawitz, 1965] D. Prawitz. *Natural Deduction - A proof theoretical study.* Almquist and Wiksell, Stockholm, 1965.

[Rivieres and Levesque, 1986] J. Des Rivieres and H.J. Levesque. The consistency of syntactical treatments of modality. In J.Y. Halpern, editor, *Theoretical Aspects of Reasoning about knowledge. Proc. 1986 conference*, pages 115–130, 1986.

[Thomason, 1980] R.H. Thomason. A note on syntactical treatments of modality. *Synthese*, 44:391–395, 1980.

[Weyhrauch, 1980] R.W. Weyhrauch. Prolegomena to a theory of Mechanized Formal Reasoning. *Artificial Intelligence. Special Issue on Non-monotonic Logic*, 13(1), 1980.

[Wilks and Biem, 1979] Y. Wilks and J. Biem. Speech acts and multiple environments. In *Proc. 6th IJCAI*, 1979.

Tractable Approximate Deduction using Limited Vocabularies

Mukesh Dalal
Rutgers University
Computer Science Department
New Brunswick, NJ 08903
dalal@cs.rutgers.edu

David W. Etherington
AT&T Bell Laboratories
AI Principles Research Department
Murray Hill, NJ 07974
ether@research.att.com

Abstract

A new approach to tractable deduction from an expressive knowledge base is presented that approximates formulae by automatically mapping them to some restricted language. Various mappings and their properties are discussed, and an anytime algorithm to compute approximations is presented. Several published approaches prove to be special instances of ours. To illustrate this, our formalism is used to formalize hierarchical knowledge bases, and to extend them by allowing negation and mutual exclusion. We believe this to be the first comprehensive theoretical framework for approximate reasoning.

1 Introduction

It is well known that reasoning with a knowledge representation system becomes more difficult as the representation language becomes more expressive [Levesque and Brachman 1985]. Reasoning is apparently intractable even for the very weak representation language of propositional logic [Cook 1971]. Since practical knowledge representation systems need more expressive representation languages [Doyle and Patil 1991], this intractability of reasoning is a serious problem.

Previous approaches to this problem have used tractable, but incomplete reasoning mechanisms. These mechanisms are often specified either algorithmically (c.f. [Allen 1983]), or by using a non-standard model theory (c.f. [Levesque 1984a]), or by using an incomplete set of inference rules (c.f. [Konolige 1986]). We present a new framework for obtaining efficient and expressive knowledge representation systems. Rather than restricting the language in which assertions and/or queries are framed, our approach restricts the *internal* representation to a certain subset, called the vocabulary, of the formulae of the original language. Every formula in the original language is then suitably mapped to one or more formulae in the vocabulary. This mapping is used to translate all the information told to, and the queries asked of, the system. We show that powerful and efficient representation and reasoning can be obtained with a variety of different vocabularies.

Mapping formulae into restricted vocabularies can produce formulae that are either stronger or weaker than the original. Consequently, our approach leads to two special kinds of approximations — those that are unsound but complete, where all, but not only, true facts are inferred; and those that are incomplete but sound, where only, but not all, true facts are inferred. These two, taken together, can provide approximate, but tractable, upper and lower bounds on the results of exact, but intractable, sound and complete reasoning.[1] Furthermore, these bounds can be refined by changing the vocabulary.

We present an anytime algorithm [Boddy and Dean 1988] that computes upper and lower bound approximations. The algorithm successively refines approximations, and can be stopped prematurely and yet yield meaningful answers. We show that it is guaranteed to provide better approximations after each refinement step.

Our approach is motivated by the Hierarchical Knowledge Bases (HKB) framework [Borgida and Etherington 1989], where the representation of input knowledge (expressed in a restricted subset of first-order logic) is restricted to instances of elements of a pre-specified hierarchy of so-called "natural" disjunctions. One such hierarchy, of pets, is shown in Figure 1, where the nodes abbreviate the natural disjunctions.[2] For instance, the natural disjunction $mouse(x) \vee gerbil(x) \vee hamster(x)$ is abbreviated as $rodent(x)$. Less natural disjunctions — for instance, $turtle(x) \vee dog(x)$ — are not directly represented in the hierarchy.

Any formula of the input language told to the system is approximated by the strongest natural disjunction(s) equivalent to or weaker than it. For instance, $turtle(joe) \vee dog(joe)$ is approximated as $carnivore(joe)$. Queries to the system are similarly approximated before being answered. If the hierarchy used to obtain such approximations is sparse (relative to the

[1] Our approach can also provide other approximations that are neither sound nor complete, but applications of such approximations are less-readily apparent.

[2] The designation of some disjunctions as "natural" is made by the designer of the hierarchy as appropriate for particular applications; the term has no metaphysical significance.

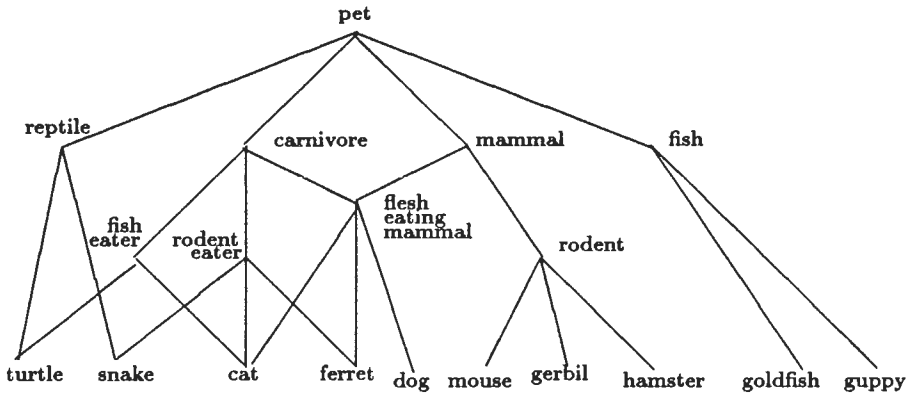


Figure 1: A simple hierarchy of pets

complete lattice of possible disjunctions), it is possible to improve the time complexity of representation and query-answering from linear to logarithmic in the size of the knowledge base [Borgida and Etherington 1989].

Our framework generalizes HKB and some other previous approaches and provides a theoretical basis for them. Results about specific formalisms then fall out as corollaries to theorems derived in the general framework. The framework can also be used to generate new mechanisms for approximate reasoning. We illustrate this by presenting instances of the framework that eliminate three of the major limitations of HKB, while maintaining its asymptotic tractability.

The rest of the paper is organized as follows. In the next section, we define the notion of vocabularies and their use in approximating formulae. In Section 3, we extend the notion of approximation to theories. Next, we define various kinds of approximate entailment and use them to obtain approximate answers to queries posed to a knowledge base. In Section 5, we present an anytime algorithm to compute approximations. We show an application of our framework, extending the expressiveness of HKBs, in Section 6. Finally, we discuss some other related research. Given the limited space available here, we relegate detailed discussion and proofs of various results to [Dalal and Etherington 1992].

2 Vocabularies and Approximations

Let $\mathcal{L}$ be any language that has a notion of formula[3], and a notion of entailment, $\models$, which is a relation between theories and formulae. The entailment relation is required to be monotonic, i.e., for any two theories Γ and Γ' and any formula ψ, if $\Gamma \models \psi$ and $\Gamma \subseteq \Gamma'$ then $\Gamma' \models \psi$. Most common logical languages — for instance, classical propositional logic and first-order predicate calculus — have monotonic entailment relations. For any theories Γ and Σ and any formula ψ, we write $\Gamma \models_\Sigma \psi$ iff $\Gamma \cup \Sigma \models \psi$, and write $\Gamma \models \Sigma$ iff for each formula $\psi \in \Sigma$, $\Gamma \models \psi$. We often abbreviate the singleton theory $\{\psi\}$ by ψ. Given this notation, we can turn to the definition of vocabularies and approximations.

Definition 1 *A* vocabulary, *V, is any (possibly infinite) subset of $\mathcal{L}$.* ■

Intuitively, the vocabulary consists of those formulae that are represented accurately. Other formulae in the language are approximated to formulae in the vocabulary, perhaps with some loss or gain of information. For example, in the HKB case [Borgida and Etherington 1989], the vocabulary is the set of all atomic formulae whose predicates are included in the hierarchy.

Any formula ψ splits the vocabulary into three parts — the formulae that entail ψ, the formulae entailed by ψ, and the others. Intuitively, the set of formulae in the vocabulary that entail ψ provides an upper-bound (stronger) approximation of ψ. Similarly, the set of formulae in the vocabulary entailed by ψ provides a lower-bound (weaker) approximation of ψ. These two approximations of ψ are called its strengthening and weakening, respectively.

Definition 2 *Let V be a vocabulary, and let Σ be a set of formulae. For any formula ψ in $\mathcal{L}$, its* weakening *$\lfloor\psi\rfloor_{V,\Sigma}$ is the set $\{\alpha \in V \mid \psi \models_\Sigma \alpha\}_\wedge$; its* strengthening *$\lceil\psi\rceil_{V,\Sigma}$ is the set $\{\alpha \in V \mid \alpha \models_\Sigma \psi\}_\vee$.* ■ [4]

The set Σ can be thought of as containing definitions (see [Brachman and Schmolze 1985]) — a set of formulae that define the terms used in the system. For example, a "parent" can be defined as "a person who has at least one child".

The subscript "$\vee$" on strengthenings prefigures the way they will be used in determining approximate entailment (Section 4). Since a formula's strengthening contains every formula that entails it, we treat strengthenings as limited disjunctions (the disjuncts do not interact). This captures the weakest strengthening possible given the vocabulary. Similarly, weakenings are treated as limited conjunctions (subscripted by "$\wedge$"), to capture the strongest possible weakening.

A weakening may be more concisely represented by a smaller set of formulae, by removing all those formulae

[3] In this paper, we assume that all formulae are closed, i.e., all variables in a formula are bound. A theory is a (possibly infinite) set of formulae.

[4] We omit the subscripts V and Σ when they are obvious from the context.

that are strictly weaker than some other formula in it. A strengthening may similarly be made concise by removing strictly stronger formulae. The complete weakening or strengthening can be easily regenerated from this concise representation.

As an example, consider the propositional language over the alphabet $\{a, b, c\}$. Let the vocabulary be the set $\{a, b, a \wedge b \wedge c, a \vee b \vee c\}$, and the set of definitions be empty. The formula a's weakening is $\{a, a \vee b \vee c\}_\wedge$; its strengthening is $\{a, a \wedge b \wedge c\}_\vee$ — both of which can be concisely represented as $\{a\}$. $a \vee b$'s weakening is $\{a \vee b \vee c\}_\wedge$. Its strengthening is $\{a, b, a \wedge b \wedge c\}_\vee$, which can be concisely represented as $\{a, b\}_\vee$.

Interactions among formulae are not considered while generating the approximations. Thus, given the previous vocabulary, neither a nor b belong to the strengthening of $a \wedge b$, since neither individually entails $a \wedge b$. Since formulae in an approximation do not interact, the weakening $\{a, b\}_\wedge$ is different from the weakening $\{a \wedge b\}_\wedge$.

In this paper, we assume that any tests of the form $\psi \models_\Sigma \alpha?$ and $\alpha \models_\Sigma \psi?$, where $\psi \in \mathcal{L}$ and $\alpha \in V$, are tractable. This assumption is satisfied when the vocabulary and the definitions are not very rich, and no individual formula, ψ, in the language $\mathcal{L}$ is very complex. This does not necessarily limit the expressiveness of $\mathcal{L}$, since theories may contain many such formulae. Our framework can be extended to the case when this assumption does not hold [Dalal and Etherington 1992].

3 Approximating Theories

The notion of approximation can be extended to theories. As in the case of formulae, the strengthening of a theory should contain exactly those formulae of the vocabulary that entail the theory. Recall that a formula entails a theory iff it entails each formula in the theory. Thus, the strengthening of a theory is obtained by intersecting the strengthenings of the formulae in it.

The corresponding extension of the notion of weakening to theories should contain all formulae of the vocabulary that are entailed by the theory. We call this the *direct weakening* of the theory. However, determining whether any particular formula in the vocabulary is entailed by a theory may be intractable, or even undecidable.

Another approach to weakening a theory is to suitably combine the weakenings of its individual formulae. Since individual formulae are usually simpler than the entire theory, this is likely to be more efficient. Since the theory entails every formula that is entailed by some formula in the theory, such a weakening of the theory can be obtained by taking the union of the weakenings of the formulae in it. This is called the *parallel weakening*.

Constructing the parallel weakening of a theory neglects the interactions among the formulae in the theory. Allowing for limited interaction among formulae yields a more accurate weakening of the theory (i.e., a tighter lower bound). Formulae can still be weakened individually, but in some sequential order. The theory is weakened one formula at a time, with the weakening obtained at each stage used in the generation of the weakening of the next formula. Formally, the weakening at any stage can be treated as part of the set of definitions used to obtain the weakening of the next formula. This is called a *sequential weakening* of the theory.

Sequential weakening depends on the sequence in which the formulae of the theory are weakened. This dependency can be removed by iterating over the sequence of formulae until a fixed point is reached. This is called the *iterative weakening* of the theory.

Definition 3 *For any vocabulary V, any set of definitions Σ, and any theory Γ of formulae in $\mathcal{L}$:*

1. *the* strengthening $\lceil\Gamma\rceil_{V,\Sigma}$ *is the set* $\{\psi \in V \mid \psi \models_\Sigma \Gamma\}_\vee$;
2. *the* direct weakening $\lfloor\Gamma\rfloor^d_{V,\Sigma}$ *is the set* $\{\psi \in V \mid \Gamma \models_\Sigma \psi\}_\wedge$;
3. *the* parallel weakening $\lfloor\Gamma\rfloor^p_{V,\Sigma}$ *is the set* $\cup_{\psi\in\Gamma}\lfloor\psi\rfloor_{V,\Sigma}$;
4. *the* sequential weakening $\lfloor\Gamma\rfloor^q_{V,\Sigma}$ *is defined recursively by:*

$$\lfloor\{\}\rfloor^q_{V,\Sigma} = \lfloor\{\}\rfloor_{V,\Sigma}$$

$$\lfloor\{\psi \mid \Omega\}\rfloor^q_{V,\Sigma} = \lfloor\Omega\rfloor^q_{V,\Sigma\cup\lfloor\psi\rfloor_{V,\Sigma}}$$

where the formulae in the theory are assumed to be sequentially ordered, and $\{\psi \mid \Omega\}$ denotes the sequence obtained by adding ψ at the beginning of the sequence Ω.

5. *the* iterative weakening $\lfloor\Gamma\rfloor^i_{V,\Sigma}$ *is defined to be* $\lfloor\Gamma^*\rfloor^q_{V,\Sigma}$, *where Γ^* is the infinite sequence obtained by repeating Γ.*[5] ■

Lemma 1 shows that iterative weakening is independent of the particular sequence of formulae in the theory. In [Dalal and Etherington 1992], we show that the iterative weakening of a finite theory can be obtained in only a finite number of iterations.

Lemma 1 *For any vocabulary V, and any set of definitions Σ, if Γ_1 and Γ_2 are two theories in $\mathcal{L}$ containing the same formulae, possibly ordered differently, then* $\lfloor\Gamma_1\rfloor^i_{V,\Sigma} = \lfloor\Gamma_2\rfloor^i_{V,\Sigma}$. ■

The following example shows that the various weakenings defined above are distinct. Consider a propositional language. Let the vocabulary be the set of all atoms, and let the theory Γ be $\{a \rightarrow b, a, a \rightarrow c, d \vee e, d \rightarrow f, e \rightarrow f\}$. The parallel weakening of Γ is $\{a\}_\wedge$, its sequential weakening (assuming the above sequence) is $\{a, c\}_\wedge$, its iterative weakening is $\{a, b, c\}_\wedge$, and its direct weakening is $\{a, b, c, f\}_\wedge$. For this restricted vocabulary, the strengthening of Γ is $\{\ \}_\vee$, which amounts to an empty disjunction, and hence is logically unsatisfiable.[6]

Consider another theory, $\{a \wedge (b \rightarrow c), b \wedge (a \rightarrow d)\}$. It can be verified that for any sequence of formulae in this

[5] Γ^* should satisfy an ordering property similar to diagonalization — for any formula $\psi \in \Gamma$, and any number $k \in N$, the kth occurrence of ψ in Γ^* should fall within a finite initial segment.

[6] Similarly, some theories will be weakened to $\{\ \}_\wedge$, which amounts to an empty conjunction, the trivially satisfiable weakening.

theory, the sequential weakening is different from the iterative weakening. Theorem 1 shows that the four types of weakenings are strictly ordered in terms of strength.

Theorem 1 *For any vocabulary V, any set of definitions Σ, and any theory Γ in $\mathcal{L}$, $\lfloor\Gamma\rfloor^p_{V,\Sigma} \subseteq \lfloor\Gamma\rfloor^q_{V,\Sigma} \subseteq \lfloor\Gamma\rfloor^i_{V,\Sigma} \subseteq \lfloor\Gamma\rfloor^d_{V,\Sigma}$.* ■

Though direct weakening is the most accurate, it is also the most expensive to construct. Constructing other weakenings is likely to be more tractable, since it only requires considering interactions between formulae in the vocabulary and definitions, rather than the entire language $\mathcal{L}$. However, for sequential and iterative weakening to be tractable, reasoning within the vocabulary and the definitions must be tractable.

4 Approximate Entailment

We have seen several ways to approximate formulae and theories. These notions can be used to determine whether a theory approximately entails a given formula. The basic idea is to reduce the question to one involving approximations of the theory and the formula, rather than using the theory or the formula directly. Depending on which types of approximation are used, various approximate entailment relations can be defined:

Definition 4 *For any vocabulary V, any set of definitions Σ, any theory Γ, any formula ψ in the language $\mathcal{L}$, and any x in $\{d, p, q, i\}$:*

1. $\Gamma \mid\!\approx^{xw}_{V,\Sigma} \psi$ *iff* $\forall\alpha \in \lfloor\psi\rfloor_{V,\Sigma}\ \exists\beta \in \lfloor\Gamma\rfloor^x_{V,\Sigma}\ (\beta \models \alpha)$;
2. $\Gamma \mid\!\approx^{xs}_{V,\Sigma} \psi$ *iff* $\exists\alpha \in \lceil\psi\rceil_{V,\Sigma}\ \exists\beta \in \lfloor\Gamma\rfloor^x_{V,\Sigma}\ (\beta \models \alpha)$;
3. $\Gamma \mid\!\approx^{ss}_{V,\Sigma} \psi$ *iff* $\forall\beta \in \lceil\Gamma\rceil_{V,\Sigma}\ \exists\alpha \in \lceil\psi\rceil_{V,\Sigma}\ (\beta \models \alpha)$;
4. $\Gamma \mid\!\approx^{sw}_{V,\Sigma} \psi$ *iff* $\forall\alpha \in \lfloor\psi\rfloor_{V,\Sigma}\ \forall\beta \in \lceil\Gamma\rceil_{V,\Sigma}\ (\beta \models \alpha)$. ■

The first and second superscripts denote the type of approximation used on the theory and the formula, respectively. Note that the formulae in the two approximations are compared pairwise, and that the formulae within an approximation do not interact. Theorem 2 shows that some of these approximate entailment relations are related to exact entailment.

Theorem 2 *For any vocabulary V, any set of definitions Σ, any theory Γ, and any formula ψ in the language $\mathcal{L}$, if Γ approximately entails ψ using any entailment in the SOUND column of the table given in Figure 2 then $\Gamma \models \psi$; if $\Gamma \models \psi$ then Γ approximately entails ψ using any entailment in the COMPLETE column.* ■

In other words, the approximate entailments in the first column are weaker than $\models$, while those in second column are stronger than $\models$. The other three entailments, $\mid\!\approx^{xw}_{V,\Sigma}$ for x in $\{p, q, i\}$, are neither stronger nor weaker than $\models$.

Various approximate entailments can also be combined to obtain more accurate approximations. For instance, the entailment defined as: $\Gamma \mid\!\approx \psi$ iff $\Gamma \mid\!\approx^{ss}_{V,\Sigma} \psi$ and $\Gamma \mid\!\approx^{dw}_{V,\Sigma} \psi$, is complete and is more accurate (i.e., weaker) than either of the two entailments used to define it.

SOUND	COMPLETE
$\mid\!\approx^{ps}_{V,\Sigma}$	$\mid\!\approx^{sw}_{V,\Sigma}$
$\mid\!\approx^{qs}_{V,\Sigma}$	$\mid\!\approx^{ss}_{V,\Sigma}$
$\mid\!\approx^{is}_{V,\Sigma}$	$\mid\!\approx^{dw}_{V,\Sigma}$
$\mid\!\approx^{ds}_{V,\Sigma}$	

Figure 2: Distinguished approximate entailments

At first, it may appear odd that the soundness and completeness of the various entailment relations are independent of the vocabulary chosen. Considering the case of the trivial (empty) vocabulary may clarify the intuitions involved. If V is empty, then the strengthening of any theory is $\{\ \}_\vee$, while all of its weakenings are $\{\ \}_\wedge$. Thus, none of the approximate entailments in the SOUND column hold, while all the other approximate entailments do. On the other hand, if $V = \mathcal{L}$, then the strengthening and direct weakening of any theory (or formula) are equivalent to the theory (formula) itself, so $\mid\!\approx^{sw}_{V,\Sigma}$, $\mid\!\approx^{ss}_{V,\Sigma}$, $\mid\!\approx^{dw}_{V,\Sigma}$, and $\mid\!\approx^{ds}_{V,\Sigma}$ all reduce to $\models$. Intuitively, the entailments $\mid\!\approx^{ss}_{V,\Sigma}$ and $\mid\!\approx^{dw}_{V,\Sigma}$ are complete for any vocabulary, since any gain or loss of information in Γ and ψ compensate for each other.

It can easily be seen from this discussion how the richness of the vocabulary is directly related to the cost of approximate reasoning, and inversely related to its accuracy.

4.1 Approximating Answers to Queries

Consider a knowledge base KB (a set of formulae) and a query Q (a formula). Viewing the knowledge base functionally [Levesque 1984a], $Ask(KB, Q)$ should return "yes" if $KB \models Q$; and "unknown" otherwise.

The classical entailment relation $\models$ is generally used to determine the correct answer to a query. However, approximate answers can be obtained by using the approximate forms of entailment defined above. If the approximate entailment relation used is sound, then all the approximate "yes" answers are correct. On the other hand, all the approximate "unknown" answers are correct if the chosen approximate entailment relation is complete. Thus, depending on the requirements that the answers need to satisfy, an appropriate notion of approximate entailment may be selected for query answering.

In general, the approximation of KB can be computed in advance, so the cost of approximation can be amortized over many queries. Furthermore, when new facts are told to KB, the old weakening can continue to be used while the new approximations are being computed, since any fact entailed by the old weakening will continue to hold under the updated version. Similarly, when facts are deleted every fact entailed by the new strengthening will hold under the old version. Hence, existing approximations can sometimes continue to be used while up-

dates are processed, although the bounds they provide may not be optimally tight. Another encouraging result is that, for either insertions or deletions, both the revised strengthening and weakening can be computed incrementally, starting from the existing approximations [Dalal and Etherington 1992].

5 Computing Approximations

We now turn to the question of computing approximations. For the purpose of this section, we assume that the vocabulary V is fixed and finite. The definitions Σ are also assumed to be fixed. We show that a formula ψ can be approximated by appropriately combining the answers to questions of the form "does $\psi \models_\Sigma \alpha$?", where α is some formula in the vocabulary. Depending on the structure of the vocabulary V, the number of such questions may vary from logarithmic to linear in the size of V.

We recall a well-known result from Algebra that allows us to represent the vocabulary in a more ordered way, such that some form of binary search can be applied. A *poset* (partially ordered set) $(A, \preceq)$ is called a *chain* if each element of A is related to every other element. A subset $B \subseteq A$ is called an *antichain* if no two elements of B are related to each other.

Theorem 3 (Dilworth) *Any poset can be expressed as a union of $|B|$ disjoint chains, where B is a longest antichain in the poset.* ■

Consider the relation $\preceq$ on V defined so that $\alpha \preceq \beta$ iff $\beta \models_\Sigma \alpha$. Notice that $(V, \preceq)$ is a poset, and let k be the size of any largest antichain in V. By Dilworth's theorem, V can be expressed as a union of k disjoint chains. Let $[V_1, \ldots, V_k]$ be a sequence of chains obtained by some such decomposition. We add two special formulae — $\mathbf{t}$ (true) and $\mathbf{f}$ (false) — to each of the otherwise disjoint chains; and extend $\preceq$ so that $\mathbf{t} \preceq \alpha \preceq \mathbf{f}$ for every formula α in the vocabulary. Note that $\mathbf{t}$ and $\mathbf{f}$ are not required to be in the vocabulary, V.

Definition 5 *A* frontier, *α, is a sequence $[\alpha_1, \ldots, \alpha_k]$ of formulae such that for each $i = 1 \ldots k$, $\alpha_i \in V_i$. α_i is called the* ith component *of the frontier.* ■

Approximations can be concisely represented using frontiers. A frontier α represents the weakening of a formula ψ iff for all i, α_i is the (logically) strongest formula in the chain V_i such that $\psi \models_\Sigma \alpha_i$. Similarly, α represents the strengthening of a formula ψ iff for all i, α_i is the (logically) weakest formula in the chain V_i such that $\alpha_i \models_\Sigma \psi$. In such a representation, approximations of any formula contain exactly k formulae of $V \cup \{\mathbf{t}, \mathbf{f}\}$.

The frontier representing an approximation of a formula can be computed by doing k binary searches. For any i, the ith component of the frontier is obtained by doing a binary search on the chain V_i. This provides an algorithm to compute approximations of any formula.

Theorem 4 *Given a finite vocabulary and a set of definitions, the algorithm sketched above correctly computes the approximations of any formula.* ■

Binary search among n items can be done using $O(\log n)$ pairwise comparisons. Each comparison here corresponds to a call of the form $\psi \models_\Sigma \alpha$? (or $\alpha \models_\Sigma \psi$?). Assume that each such call takes time m (a function of the particular ψ and Σ). Thus, obtaining the result from a chain V_i requires time $O(m \log |V_i|)$. Since we need results from all chains, computing an approximation takes time $O(m \sum_{i=1}^{k} \log |V_i|)$. In the worst case, when all the chains are of the same size, it takes time $O(mk \log(|V|/k))$ to compute either a weakening or a strengthening. Recall our assumption from Section 2 that any test of the form $\psi \models_\Sigma \alpha$? and $\alpha \models_\Sigma \psi$?, where $\psi \in \mathcal{L}$ and $\alpha \in V$, is tractable.

5.1 Anytime Guarantees

The algorithm described above has an interesting property — it can be stopped at any time before completion to obtain a "reasonable" approximation of the exact frontier. Moreover, certain guarantees can be made about the quality of the approximation. For this, we need to define a metric to describe the quality of a partial answer.

Definition 6 *The* distance *between two elements of a chain is the number of elements (strictly) stronger than one but weaker than the other. The* distance *between two frontiers is the sum of the distances between their corresponding components.* ■

Intuitively, the distance between two frontiers measures how much latitude or "slack" the vocabulary allows for expression of concepts between them. Thus, the larger the distance between two frontiers, the less tightly they constrain what lies between them, and the more room for error.

Let us assume that d calls to the recursive binary search procedure have been made before the algorithm is prematurely stopped. We assume that these calls have been made greedily (i.e., make the call that has the longest chain left to search). Let $(\alpha_{x_i}, \ldots, \alpha_{y_i})$ be the subchain given to the last (incomplete) call for each chain V_i. We claim that the frontiers $X = [\alpha_{x_1}, \ldots, \alpha_{x_k}]$ and $Y = [\alpha_{y_1}, \ldots, \alpha_{y_k}]$ are good partial answers for the weakening and strengthening, respectively. Thus, the quality of the approximate answer is guaranteed to improve if the algorithm is allowed to run longer. Lemma 2 makes this more precise.

Lemma 2 *If the algorithm to compute the weakening (or strengthening) of a formula is stopped after d recursive calls of binary search, then the distance of X and Y from the weakening or strengthening, respectively, is at most $O(|V| 2^{-d/k})$.* ■

6 An application: Extending HKBs

Our framework can be used to generate new mechanisms for approximate reasoning. We illustrate this by presenting instances of this framework that eliminate three major limitations of HKB [Borgida and Etherington 1989], while maintaining its tractability. Other mechanisms for approximate reasoning can be similarly obtained by instantiating the framework.

Recall that HKB represents information using a hierarchy of naturally-occurring disjunctions, such as that

shown in Figure 1. When a positive formula is told to the HKB system, the clauses in the hierarchy that are subsumed (i.e., logically entailed) by the formula are made true. When a positive formula is asked as a query, there are two types of answers. Lower-bound answers are obtained by querying each maximal node in the hierarchy subsumed by the query, in turn — the answer is true iff any of them is true. Upper-bound answers are obtained by querying each minimal node in the hierarchy that subsumes the query — the answer is true iff all of them are true.[7]

We first show that the HKB framework is a special case of the framework presented here. Let H be the set of predicates in the hierarchy. The language $\mathcal{L}$ contains a restricted subset of positive formulae built from the predicates in H. The vocabulary V consists of all formulae of the form $P(x)$, where P is a predicate in H and x is an individual. The definitions Σ consist of all formulae of the form $\forall x(P(x) \leftrightarrow P_1(x) \vee \ldots \vee P_n(x))$, where $P_1, \ldots, P_n$ are all the immediate children of a node P in the hierarchy.

Theorem 5 *Any formula ψ in $\mathcal{L}$ told to the HKB system is represented by its weakening $\lfloor\psi\rfloor_{V,\Sigma}$. For any theory in $\mathcal{L}$, the four kinds of weakening are identical. Given any positive query ψ to a knowledge base Γ, the upper-bound answer is true iff $\Gamma \mid\!\approx^{dw}_{V,\Sigma} \psi$. Given a clausal query ψ to a knowledge base Γ, the lower-bound answer is true iff $\Gamma \mid\!\approx^{ds}_{V,\Sigma} \psi$.*[8] ■

Since all four kinds of weakening are identical, a theory can be weakened by weakening its formulae independently. In [Dalal and Etherington 1992], we show that certain special properties of the vocabulary allow tractable weakening of individual formulae. Since $\mid\!\approx^{ds}_{V,\Sigma}$ is always sound, it follows that lower-bound answers are sound. Also, since $\mid\!\approx^{dw}_{V,\Sigma}$ is always complete, it follows that upper-bound answers are complete. Thus, the results in [Borgida and Etherington 1989] are special cases of our results.

In [Dalal and Etherington 1992], we show new instances of our framework that avoid three major limitations of HKB, while maintaining its asymptotic tractability:

Negation: We allow representation of, and reasoning with, limited forms of negative information. For instance, if all fish-eaters are assumed to be either turtles or cats, then the system can be told $fish\text{-}eater(joe)$ and $\neg turtle(joe)$, leading it to infer $cat(joe)$. Formally, the language $\mathcal{L}$ is extended to include conjunctions of negative literals using the predicates in H; and the vocabulary V is extended to include $\neg P(x)$, where P is a predicate in H and x is an individual. In this case, the iterative weakening of any theory is identical to its direct weakening. Also, if all the negative formulae are ordered before all the positive formulae then the sequential weakening is also identical to the iterative weakening. However, if the vocabulary is extended to also allow some negative clauses, like $\neg mouse(stan) \vee \neg cat(stan)$, then computing the sequential weakening becomes CoNP-hard. Thus, the above extension appears to be the best that can be done without losing tractability.

Mutual Exclusion: We allow certain nodes in the hierarchy to be declared to be mutually exclusive, i.e., having no common instances. For example, all the leaves in the pet hierarchy could be declared to be mutually exclusive, making it possible to infer $cat(joe)$ from $fish\text{-}eater(joe)$ and $rodent\text{-}eater(joe)$. Formally, two types of mutual exclusion can be allowed. In one form, all the children of a node can be declared to be (pairwise) mutually exclusive; in the other form, all the leaf-descendants of a node can be declared to be mutually exclusive. Both forms are represented by adding more formulae to the set of definitions Σ of HKB. In the latter case, if P is the node in question, then the definitions include formulae of the form $\forall x(C(x) \rightarrow \neg D(x))$, for each pair C and D of distinct leaf nodes under P. In the former case, the definitions include formulae of the form $\forall x(C(x) \rightarrow \neg D(x))$, for each pair C and D of distinct children of P.

Tighter Bounds: Given the hierarchy of Figure 1 and the information $cat(joe)$, the original HKB theory would simplistically specify "yes" as the upper-bound answer to the query $cat(joe) \wedge (turtle(joe) \vee dog(joe))$. We can get better upper-bound answers by extending the hierarchy to include some conjunctive formulae. We then use the entailment relation obtained by intersecting $\mid\!\approx^{dw}$ and $\mid\!\approx^{ss}$: since both of these are complete, their intersection is also complete. Moreover, this yields an answer that is more accurate than the upper-bound answer provided by the original HKB, which uses $\mid\!\approx^{dw}$ alone.

7 Related Research

Building on the intuitions in [Hobbs 1985], Imielinski [1987] proposes two mechanisms to abstract a knowledge base to obtain a potentially-simpler knowledge base that can be used to answer queries posed to the original knowledge base. In [Dalal and Etherington 1992], we show that both abstractions are special cases of our framework. Moreover, a new notion of abstraction can be developed using the ideas we have outlined.

Fagin and Halpern [1985] define a logic of general awareness, in which the beliefs of an agent are restricted to a pre-specified subset of formulae. This "awareness set" is closely related to the concept of vocabulary introduced above, and the limited beliefs entailed by a theory correspond to the direct weakening of the theory. However, there are no notions corresponding to the other kinds of weakening nor to strengthening.

Selman and Kautz [1991] present a system in which formulae are compiled into a restricted language that allows efficient inference. Though their motivation is very similar to ours, the techniques differ. They approximate each theory by an upper- and a lower-bound in

[7] Borgida and Etherington also use a form of the generalized closed-world assumption [Minker 1982] to obtain "no" answer in certain cases. We ignore this aspect.

[8] For queries containing both $\wedge$ and $\vee$, the lower-bound answer is determined by applying the following recursive rules until a clausal query is obtained: $\psi \wedge \omega$ is true iff ψ is true and ω is true, and $\psi \vee \omega$ is true iff ψ is true or ω is true.

a tractable target language. Their upper bound corresponds to the direct weakening in our framework. However, they have no notions corresponding to the other more tractable types of weakening, and their notion of strengthening is substantially different. Furthermore, they do not approximate queries.

There have been other approaches to the problem of the intractability of reasoning, in various other contexts. Some of them ([Levesque 1984b], [Allen 1983], etc.) are special cases of the generalization of this framework presented in [Dalal and Etherington 1992].

8 Conclusions

We presented a framework for approximate reasoning using a limited vocabulary, and showed that it improves on many previous approaches, as well as facilitating the development of specific tractable reasoning techniques. We believe this is the first comprehensive theoretical framework for approximate reasoning.

We presented an anytime algorithm that, for finite vocabularies, computes weakenings and strengthenings of formulae. These approximations can be used to determine whether a theory approximately-entails a formula. In [Dalal and Etherington 1992], we extend our treatment to infinite vocabularies, where approximate entailment is determined without explicitly computing entire approximations.

One limitation of this framework is the assumption that the $\models$ tests between pairs of individual formulae, taken one from the language and the other from the vocabulary, are tractable. In general, these tests may be intractable for complex formulae. We have generalized the framework presented here [Dalal and Etherington 1992]. In the generalized version, the approximation of a formula is obtained by combining the approximations of its sub-formulae, which can reduce the complexity significantly.

Acknowledgments

We would like to thank our colleagues Alex Borgida, Henry Kautz, Bart Selman, James Crawford, Kumar Vadaparty and Ringo Ling for helpful discussions and comments on the earlier drafts of this paper. We also wish to thank AT&T Bell Laboratories for support for this research.

References

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

Boddy, M. and Dean, T. 1988. Solving time dependent planning problems. Technical report, Dept. of Computer Science, Brown University.

Borgida, A. and Etherington, D. 1989. Hierarchical knowledge bases and efficient disjunctive reasoning. In Brachman, R., Levesque, H., and Reiter, R., editors, *Proceedings First International Conference on Principles of Knowledge Representation and Reasoning*, pages 33–43. Morgan Kaufmann.

Brachman, R. and Schmolze, J. 1985. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9:171–216.

Cook, S. 1971. The complexity of theorem proving procedures. In *Proceedings Third Annual ACM Symposium on the Theory of Computing*, pages 151–158.

Dalal, M. and Etherington, D. W. 1992. A general framework for approximating deduction. In preparation.

Doyle, J. and Patil, R. 1991. Two theses of knowledge representation: language restrictions, taxanomic classification, and the utility of representation services. *Artificial Intelligence*, 48(3):261–297.

Fagin, R. and Halpern, J. 1985. Belief, awareness and limited reasoning. In *Proceedings Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 491–501.

Hobbs, J. 1985. Granularity. In *Proceedings Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 432–435.

Imielinski, T. 1987. Domain abstraction and limited reasoning. In *Proceedings Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 997–1003, Milan.

Konolige, K. 1986. *A deduction model of belief.* Reasearch Notes in AI. Pitman, London and Morgan Kaufmann, California.

Levesque, H. and Brachman, R. 1985. A fundamental tradeoff in knowledge representation and reasoning (revised version). In Brachman, R. and Levesque, H., editors, *Readings in Knowledge Representation*, pages 41–70. Morgan Kaufmann, Los Altos, California.

Levesque, H. 1984. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23(2):155–212.

Levesque, H. 1984. A logic of implicit and explicit belief. *Proceedings National Conference on Artificial Intelligence (AAAI-84)*, pages 198–202.

Minker, J. 1982. On indefinite databases and the closed-world assumption. In Loveland, D., editor, *Proceedings Sixth International Conference on Automated Deduction, Springer-Verlag Lecture Notes in Computer Science 138*, pages 292–308.

Selman, B. and Kautz, H. 1991. Knowledge compilation using Horn approximations. In *Proceedings Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 904–909.

A Formal Analysis of Solution Caching

Vinay K. Chaudhri*
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4
vinay@ai.toronto.edu

Russell Greiner†
Siemens Corporate Research
Princeton, NJ 08540
greiner@learning.siemens.com

Abstract

Many inference management systems store and maintain the conclusions found during a derivation process in a form that allows these conclusions to be used during subsequent derivations. As this approach, called "solution caching", allows the system to avoid repeating these derivations, it can reduce the system's overall cost for answering queries. Unfortunately, as there is a cost for storing these conclusions, it is not always desirable to cache every solution found — this depends on whether the savings achieved by performing fewer inference steps for these future queries exceeds the storage overhead incurred. This paper formally characterizes this tradeoff and presents an efficient algorithm, FOCL, that produces an optimal caching strategy: *i.e.*, given an inference graph of a knowledge base, anticipated frequencies of queries and updates of each node in this graph, and various implementation-dependent cost parameters, FOCL determines which of these nodes should cache their solutions to produce a system whose overall cost is minimal. The paper also presents empirical results that indicate that a system based on FOCL can significantly outperform one based on any of the standard approaches to solution caching.

1 Introduction

A "solution caching"[1] system will store and maintain the conclusions found during a derivation process, in a form that allows the system to simply retrieve and reuse these stored solutions to answer subsequent queries. As this avoids the cost of repeating these derivations, it can significantly improve the response time for repeated queries. These savings are especially important in very large and complex knowledge bases and in applications where the response time is critical, such as real time process control. As caching does incur the cost of storing the derived conclusions, it may not be useful when the storage cost is very high or when the queries are not repeated a large number of times.

As an example, consider the knowledge base, KB_1, shown in Figure 1.[2] If we ask for all `living` objects (*i.e.*, find all `X` satisfying the `living(X)` query), the inference engine will backward chain, traversing the inference graph down to the ground facts. Here, the solutions are:

$$\Delta = \left\{ \begin{array}{lll} \text{living(slime2)} & \text{living(george)} & \text{living(john)} \\ \text{living(fred)} & \text{living(fido)} & \text{living(roxy)} \\ \text{living(apple1)} & \text{living(orange3)} & \text{living(bean7)} \end{array} \right\} \quad (1)$$

An inference management system IMS[3] that can cache its answers could then store these derived solutions, in effect forming a larger knowledge base $KB_1' \leftarrow KB_1 \cup \Delta$ that includes all of KB_1 as well as these nine propositions, Δ. If the same query `living(X)` is posed again, this IMS will find these solutions by a single lookup rather than by re-deriving them by backward chaining. Notice the IMS is spending additional time in processing the initial query to store this information, in the hope that it will save time later when addressing this same query for the second and subsequent times. Notice, however, that these cached entries can become "dirty" if the fact set is changed — *e.g.*, if we delete the literal `man(john)`, or add a new fact `dog(shep)`. Hence, we must consider how often each type of fact is updated and the cost of propagating this change to insure the cached information remains accurate.

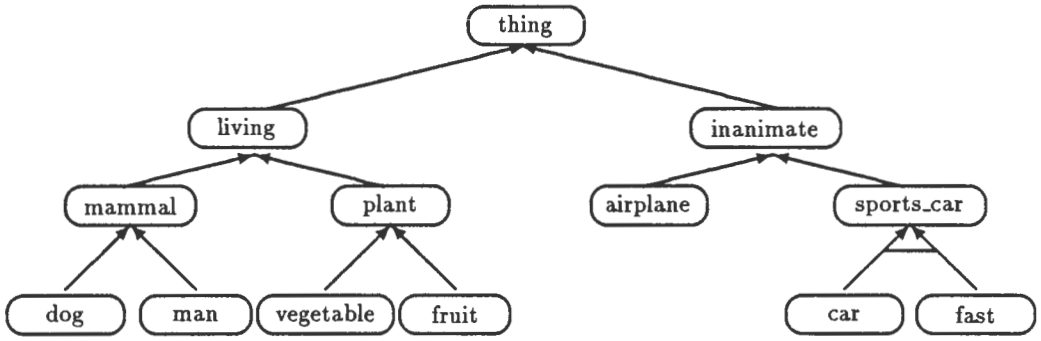
*Supported by the Department of Computer Science at the University of Toronto and by the Information Technology Research Centre of Ontario. Both authors gratefully acknowledge the help from Professors Charles Elkan and M. W. Carter, as well as from Tom Petsche, Wen-Ling Hsu and the anonymous referees.

†Much of this work was performed at the University of Toronto where it was supported by the Institute for Robotics and Intelligent Systems and by an operating grant from the National Science and Engineering Research Council of Canada.

[1] We will use the term "caching" as a shorthand for this form of "solution caching". *N.b.*, it does *not* refer to the technique of moving information from secondary to primary storage.

[2] Following PROLOG conventions, names that begin with a capital letter (*e.g.*, "`X`") are variables. Unbound variables that appear in assertions are assumed universally quantified; those in queries, existentially quantified.

[3] This IMS can be a knowledge representation system, a deductive database, a logic programming system, an object-oriented system, etc.



Rule Base

```
thing(X)     :- living(X).     thing(X)     :- inanimate(X).
living(X)    :- mammal(X).     living(X)    :- plant(X).
mammal(X)    :- dog(X).        mammal(X)    :- man(X).
plant(X)     :- vegetable(X).  plant(X)     :- vegetable(X).
inanimate(X) :- vehicle(X).    inanimate(X) :- airplane(X).
sports_car(X) :- car(X), fast(X).
```

Fact Set

```
living(slime2)    mammal(george)
man(john)         man(fred)
dog(fido)         dog(roxy)
fruit(apple1)     vegetable(bean7)
fruit(orange3)    airplane(p17)
```

Figure 1: Knowledge Base KB_1 — Ground facts + Rules

- Structure of the inference graph
- Distribution of queries and updates
- Costs of each inference step, fact-set retrieval, ...
- Incremental cost of additional storage
- Number of solutions to be cached (at each node)

Table 1: Factors Affecting Caching Performance

Our goal is to use solution caching as a way of producing an efficient IMS — one that requires a minimal amount of time to deal with the anticipated distribution of queries and updates.

There are two standard ways of dealing with solution caching. Many systems, including PROLOG [CM81], never cache any solutions. Others (*e.g.*, [Mos83]) will cache every solution found — *e.g.*, at every node in the graph shown in Figure 1. This paper shows that neither of these two simple approaches leads to an optimally efficient system, and so proposes a third approach: of selectively caching only at certain nodes. Section 2 first develops a quantitative model that formalizes the interaction amongst the different factors affecting the caching performance, summarized in Table 1. Section 3 then uses this model to define an algorithm, FOCL (for "Find Optimal Cache Label") that determines which literals should cache their solutions. Section 4 then presents a set of performance experiments to demonstrate that the optimal caching scheme produced by FOCL can significantly outperform the systems that use either obvious approach, of caching everywhere or not caching anywhere.

Due to space restrictions, this short article cannot provide a comprehensive survey of the related research; instead, we refer the interested reader to our extended paper [CG92]. That paper also discusses how our results can be used by a wide variety of systems, including knowledge representation, deductive databases, logic programming and object-oriented systems; and presents both the relevant proofs and a more comprehensive discussion of the FOCL algorithm.

2 Framework and Cost Model

This work deals with definite clause knowledge bases (*i.e.*, a set of clauses, where each clause has exactly one positive literal); we call each ground atomic literal a "fact", and each non-atomic clause, a "rule". We can arrange the rules into an inference graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{A} \rangle$, where each node $n \in \mathcal{N}$ corresponds to an atomic literal, and each hyper-arc $a \in \mathcal{A}$ corresponds to a rule, leading from (the nodes representing) its set of antecedents to (the node representing) its conclusion; see, *e.g.*, Figure 1.[4] As discussed above, given a query (*e.g.*, "`living(Y)`"), the IMS will search through the graph seeking all solutions, both those corresponding to an immediate database retrieval (which finds `living(slime2)`) as well as the solutions found by backward chaining — *i.e.*, following the various rules to their subgoals, to obtain the other eight solutions shown in Equation 1. The IMS will then return all of these answers.

The IMS may also decide to "cache" these solutions — *e.g.*, store all of the Δ set associated with the `living(X)` node. It may also store the solutions found at any intermediate node — *e.g.*, store the facts {`mammal(george)`, `mammal(john)`, `mammal(fred)`, `mammal(fido)`, `mammal(roxy)`} associated with the `mammal(X)` node, or the facts {`plant(bean7)`, `plant(apple1)`, `plant(orange3)`} associated with the `plant(X)` node, etc.

Our IMS has the option of caching at any of the nodes in the graph — if so, it will store *all* of the derived solutions associated with each selected node. As an example, the first time the IMS encounters the `living(X)` query

[4] We will often identify each node $n \in \mathcal{N}$ with its associated literal.

(perhaps as a subquery of the "higher" `thing(X)` query) it will compute the bindings shown in Equation 1. If it has decided to cache at this node, it will then store the subset of these solutions that are not already explicitly present — that is, all but `living(slime2)`. The second, and subsequent times IMS encounters this `living(X)` query, it will simply retrieve all nine solutions (the eight newly stored values, and the one originally stored) and so will not need to backward chain.

The retrieval time required to find these answers is clearly much less when these answer have been cached. There is, however, a cost to storing these solutions initially, and there is also an additional cost each time there is an update to any of the relations used in any of `living(X)`'s "children" — *i.e.*, if we add `dog(shep)`, etc. Hence, it is not obvious whether we should cache at any of the nodes.

We can formally define our task in terms of the following definition:

Defn#1. A Caching Label of an inference graph is a function which assigns a label of either "C^+" or "C^-" to each node in the graph. The label "C^+" (resp., "C^-") means that solutions should (resp., should not) be cached at this node.
We let $\mathcal{LL}(\mathcal{G})$ refer to the set of all labels for the inference graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{A} \rangle$ — *i.e.*,

$$\mathcal{LL}(\langle \mathcal{N}, \mathcal{A} \rangle) \stackrel{def}{=} \{ \mathcal{L}_\ell \,|\, \mathcal{L}_\ell : \mathcal{N} \mapsto \{C^+, C^-\} \};$$

and let $\mathcal{L}_\ell[n_k]$ refer to the value the labelling $\mathcal{L}_\ell \in \mathcal{LL}(\mathcal{G})$ assigns to the node $n_k \in \mathcal{N}$.

The Overall Cost of a caching label $\mathcal{L}_\ell$, written $E[\mathcal{L}_\ell]$, is the total cost required to perform all anticipated queries at all nodes, assuming the IMS uses the labelling $\mathcal{L}_\ell$. This value is the sum of the costs associated with each node in the graph, and includes the costs of performing all retrieving inferencing, storing and updating steps. (The particular formula for these costs will depend on the type of inference graph and cost model involved; see the next subsection, especially Equation 3.)

Optimal Cache Labelling Task:
Instance: An inference graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{A} \rangle$ with its space of cache labellings $\mathcal{LL}(\mathcal{G})$, and cost function $E : \mathcal{LL}(\mathcal{G}) \mapsto \Re$.

Problem: Identify a cache labelling $\mathcal{L}_* \in \mathcal{LL}(\mathcal{G})$ whose cost is minimal over all labellings — *i.e.*, such that
$\forall \mathcal{L} \in \mathcal{LL}(\mathcal{G})\ E[\mathcal{L}_*] \leq E[\mathcal{L}]$. □

The rest of this subsection sketchs a particular concrete cost model (*i.e.*, a specific $E[\cdot]$ function) for a particular class of inference graphs; [CG92] provides a more exact specification.[5]

[5]This paper uses a very simple model for purely pedagogical reasons; we are aware that sophisticated PROLOG systems require a much more elaborate model [Debray, personal communication]. The analysis in this paper *does* apply to those models as well; see [CG92].

We assume that the total cost of a database retrieval varies linearly with the number of solutions obtained — *e.g.*, the cost of retrieving the 2 facts matching `man(Y)` is twice the cost of retrieving the 1 fact matching `vegetable(Z)` (*viz.*, `vegetable(bean7)`). In general, it will cost "nL" to retrieve the n facts matching a proposition, where L is a constant that is independent of the particular proposition considered; *i.e.*, independent of whether the query was `man(Y)` or `vegetable(Z)`.

We assume a uniform cost for caching any proposition; *i.e.*, it costs the same to cache `dog(fido)` as to store `between(a P b)`; call this value "S". It will also cost this same amount to perform any *update* to a cache — whether by adding or deleting a literal. We likewise assume a uniform cost "R" for reducing any goal along any one rule, to that rule's (appropriately instantiated) antecedents. For example, it costs R to reduce `thing(X)` to `living(X)`; and costs the same R to reduce `sports_car(X)` to `car(X)` and `fast(X)`.

We assume, as given, the values of these parameters:

L – Cost of any one lookup, per unit fact found
R – Cost of reducing any one goal (along any one rule)
S – Cost of caching/adding/deleting any one fact

In addition, for each node n_i in the inference graph, we must know

Defn#2. $s(n_i)$ is the current number of propositions explicitly in the fact set that match the goal associated with this node. As an example based on the graph in Figure 2 (taken from the far left side of Figure 1's KB_1), $s(n_4)$ is the number of propositions in the fact set that match `dog(X)`. Assuming the associated fact set contains (all and only) the facts {`dog(fido)`, `dog(roxy)`, `mammal(lulu)`}, then $s(n_4) = 2$, $s(n_3) = 1$ and $s(n_1) = s(n_2) = 0$.

Defn#3. $d(n_i)$ is the number of direct queries posed at the node n_i. *E.g.*, we will ask the question `mammal(X)` a total of $d(n_3)$ times.

Defn#4. $u(n_i)$ is the rate of updates to the node n_i, where each update is either adding or deleting a literal to the extension of the node n_i. As an example, if we plan to add in two new literals — *e.g.*, `dog(shep)` and `dog(phydeau)` — and delete one literal `dog(fido)` over a period of time during which the number of queries was 100, then $u(n_4) = (2+1)/100 = 0.03$.

(The values of $d(\cdot)$ and $u(\cdot)$ are with respect to some interval of time; see Subsection 3.4. That subsection also discusses how to estimate the values of these parameters.)

Given the values of these implementation-dependent parameters, we can compute the cost $E[\mathcal{L}_\ell]$ of any given $\mathcal{L}_\ell \in \mathcal{LL}(\mathcal{G})$. We need the following terms:

Defn#5. Given any node $n \in \mathcal{N}$, let
$Ch(n) = \{n_i \,|\, \langle n, n_i \rangle \in \mathcal{A}\}$ refer to n's immediate children; and
$\mathcal{U}(n)$ refer to the set of nodes in the graph strictly strictly "under" n, at any depth: *i.e.*,

$$\mathcal{U}(n) = Ch(n) \cup \bigcup_{n_i \in Ch(n)} \mathcal{U}(n_i).$$

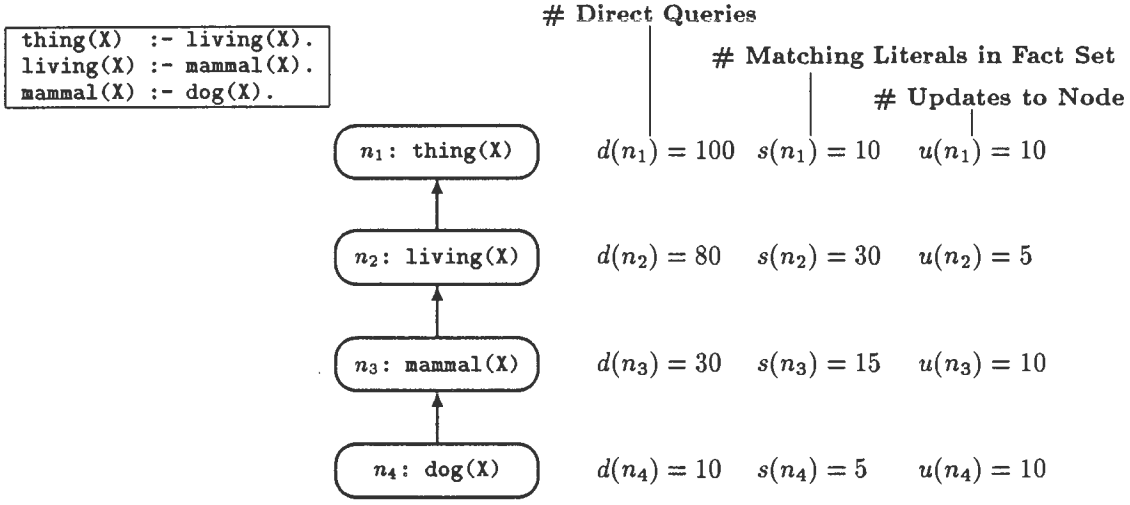


Figure 2: Knowledge Base KB_2; and Parameters

(Notice $n \notin \mathcal{U}(n)$; and if n is a leaf, then $Ch(n) = \mathcal{U}(n) = \{\}$.)

Defn#6. The Number of Indirect Queries at the node n_k with respect to a labelling $\mathcal{L}_\ell$, designated "$I_\ell(n_k)$", is the total number of queries that the user can ask at any of n_k's ancestor nodes and that will cause the inference process to retrieve values at n_k. Notice this value depends on the cache labelling.

If we cache at the parent node n_k, then the child n_{k+1} receives only one indirect query — only for the first derivation. (*E.g.*, if we cache at `living(·)`, then its child `mammal(·)` will receive only a single indirect query.) If we do not cache, the number of indirect queries that n_{k+1} receives is the sum of direct and indirect queries at the parent. Hence,

$$I_\ell(n_{k+1}) = \begin{cases} I_\ell(n_k) + d(n_k) & \text{if } \mathcal{L}_\ell[n_k] = \mathrm{C}^- \\ 1 & \text{if } \mathcal{L}_\ell[n_k] = \mathrm{C}^+ \end{cases} \quad (2)$$

As the root node n_1 does not have any parents, we have $I_\ell(n_1) = 0$ for every $\mathcal{L}_\ell$.

To illustrate this: using Figure 2, let $\mathcal{L}_{\langle----\rangle}$ denote the labelling that does not cache at any node, and $\mathcal{L}_{\langle-+--\rangle}$, the labelling that caches only at the node n_2 and nowhere else. Then the number of indirect queries at n_3 is $I_{\langle----\rangle}(n_3) = d(n_1)+d(n_2)$; and $I_{\langle-+--\rangle}(n_3) = 1$.

For any node $n \in \mathcal{N}$, define $E[\mathcal{L}_\ell, n]$ to be the cost of using the label $\mathcal{L}_\ell$ to deal with the nodes including and below n — *i.e.*, with the nodes $\{n\} \cup \mathcal{U}(n)$. Notice $E[\mathcal{L}_\ell] = E[\mathcal{L}_\ell, \langle\texttt{root}\rangle]$, where $\langle\texttt{root}\rangle$ is the root node (here n_1). The incremental cost of dealing with the node n, above the cost of its children, involves the expense of retrieving n's complete extension a total of $d(n) + I_\ell(n)$ times. If we cache, then we must add in the cost of caching the additional $I_\ell(n)$ answers after answering the query for the first time, and also the cost of returning these cached solutions during each subsequent retrieval. We also have the additional cost of the processing the subsequent updates. Hence,

$$E[\mathcal{L}_\ell, n] = \sum_{n_i \in Ch(n)} E[\mathcal{L}_\ell, n_i] + \begin{cases} [(L \cdot s(n)) + (R \cdot |Ch(n)|)](I_\ell(n) + d(n)) & \text{if } \mathcal{L}_\ell[n] = \mathrm{C}^- \\ [(L \cdot s(n)) + (R \cdot |Ch(n)|)] & \text{if } \mathcal{L}_\ell[n] = \mathrm{C}^+ \\ \quad + [d(n) + I_\ell(n) - 1] \cdot (L \cdot [s(n) + s(\mathcal{U}(n))]) & \\ \quad + S \cdot s(\mathcal{U}(n)) + S \cdot u(\mathcal{U}(n)) & \end{cases} \quad (3)$$

where $s(\mathcal{U}(n)) = \sum_{n_i \in \mathcal{U}(n)} s(n_i)$ and $u(\mathcal{U}(n)) = \sum_{n_i \in \mathcal{U}(n)} u(n_i)$. Notice the value of $E[\mathcal{L}_\ell, n]$ depends on both the labels of the nodes *below* n (as it involves $E[\mathcal{L}_\ell, n_i]$ for each $n_i \in Ch(n)$), and the labels of the nodes *above* n (as it involves $I_\ell(n)$).

The precise characterization of the cost, shown in Equation 3, is one of the important contributions of our work. Notice it extends previous work (*e.g.*, [Sel89, SJGP90]) which assumes that this cost is given and is independent of the structure of the knowledge base.

3 Optimal Labelling Algorithm

For pedagogical reasons, Subsection 3.1 first describes the FOCL algorithm for a simple class of inference graphs; Subsections 3.2 and 3.3 then discuss how FOCL generalizes to cover other classes, enabling FOCL to handle any "tree structured" inference graph; *i.e.*, any graph that includes at most one directed path between any pair of nodes. (Figure 1 is an example.) FOCL depends on various input values; Subsection 3.4 discusses ways of obtaining or estimating these values.

3.1 Using FOCL for Linear KBs

This subsection deals only with the particular class of "linear knowledge bases", where each clause can have at most one negative literal and the conclusion of at most one rule can match any given proposition. (Hence, each "rule" can have only one antecedent, meaning there are no conjunctions; and any goal can be reduced to at most one subgoal, so $|Ch(n)| \leq 1$ for all nodes n.) The graph in Figure 2 suggests such a knowledge base.

$$
\begin{array}{lcll}
P(n_4,0) & = & \{\mathcal{L}_\ell \mid \mathcal{L}_\ell[n_1]=\mathrm{C}^- \;\&\; \mathcal{L}_\ell[n_2]=\mathrm{C}^- \;\&\; \mathcal{L}_\ell[n_3]=\mathrm{C}^- \} & \langle - - - ? \rangle \\
P(n_4,1) & = & \{\mathcal{L}_\ell \mid \mathcal{L}_\ell[n_1]=\mathrm{C}^+ \;\&\; \mathcal{L}_\ell[n_2]=\mathrm{C}^- \;\&\; \mathcal{L}_\ell[n_3]=\mathrm{C}^- \} & \langle + - - ? \rangle \\
P(n_4,2) & = & \{\mathcal{L}_\ell \mid \cdot \qquad\qquad \mathcal{L}_\ell[n_2]=\mathrm{C}^+ \;\&\; \mathcal{L}_\ell[n_3]=\mathrm{C}^- \} & \langle ? + - ? \rangle \\
P(n_4,3) & = & \{\mathcal{L}_\ell \mid \qquad\qquad\qquad\quad \mathcal{L}_\ell[n_3]=\mathrm{C}^+ \} & \langle ? ? + ? \rangle
\end{array}
$$

Figure 3: Values of $P(n_i, j)$

One naïve way to find the optimal labelling is to enumerate all of the possible labellings, compute the cost of each and select the one that is minimum. This approach is not computationally feasible even for this simple class of knowledge bases, as there are $2^{|\mathcal{N}|}$ labellings.

Another approach is to label each node one at a time, by traversing the entire inference graph in one direction — either top down or bottom up. Unfortunately, the decision of whether to cache at a node depends on the cost of answering all queries at that node, which in turn depends on the labels of both the ancestor and the descendant nodes: the total number of queries that reach a node depend on which of its ancestors are cached, and the cost of obtaining the complete extension of a node will depend on which of its descendants are cached. This rules out a single traversal in either direction, as either requires quantities that would not be available. Fortunately, however, we can capture this interdependence using a dynamic programming technique to obtain a solution that is provably optimal [Nem66].

The basic idea involves two traversals of the inference graph; see Figure 4. Equation 3 shows that the value of $E[\mathcal{L}_\ell, n_k]$ depends on $E[\mathcal{L}_\ell, n_{k+1}]$, $I_{\mathcal{L}_\ell}(n_k)$ and various input parameters. Fortunately, the values of $E[\mathcal{L}_\ell, n_{k+1}]$ and $I_\ell(n_k)$ can be decoupled: given any class of labellings that share a common $I_\ell(n_k)$ value, the best labelling will be the one with the smallest $E[\mathcal{L}_\ell, n_{k+1}]$ value.

FOCL's first pass [Figure 4's Line 1] works from the root down to the leaf node (here from n_1 to n_4), partitioning the set of possible labellings into equivalence classes that share a common value of $I_\ell(n_k)$. That is, define

$$
\begin{array}{l}
P(n_k,0) = \{\ \mathcal{L}_\ell \in \mathcal{LL}(\mathcal{G}) \mid \forall\, 0 < j < k.\ \mathcal{L}_\ell[n_j] = \mathrm{C}^- \} \\
P(n_k,i) = \{\ \mathcal{L}_\ell \in \mathcal{LL}(\mathcal{G}) \mid \forall\, i < j < k.\ \mathcal{L}_\ell[n_j] = \mathrm{C}^- \;\&\; \mathcal{L}_\ell[n_i] = \mathrm{C}^+ \} \quad \text{for } i = 1..(k-1)
\end{array} \tag{4}
$$

Notice $P(n_i, j)$ is a set of labels. As an example, $P(n_1, 0) = \mathcal{LL}(\mathcal{G})$ is the set of all labellings.

Figure 3 describes the values of $P(n_4, j)$ for the allowed values of j. Its right side encodes each $P(n_i, j)$ as a sequence of the form $\langle \pm_1, \pm_2, \ldots \pm_k \rangle$ where, for each $\mathcal{L}_\ell \in P(n_i, j)$, $\pm_m = +$ means $\mathcal{L}_\ell[n_m] = \mathrm{C}^+$, $\pm_m = -$ means $\mathcal{L}_\ell[n_m] = \mathrm{C}^-$, and $\pm_m = ?$ means $\mathcal{L}_\ell[n_m]$ is arbitrary. In general,

$$
P(n_i, j) \equiv \langle \underbrace{? ? \cdots ?}_{\text{irrelevant}} \overset{i^{th}}{\downarrow} + \underbrace{- - \cdots -}_{\text{all } -\text{'s}} \overset{j^{th}}{\downarrow} \underbrace{? ? \cdots ?}_{\text{irrelevant}} \rangle
$$

for $i = 1..(j-1)$. Notice $P(n_i, j)$ does not restrict labels on the basis of their values for nodes n_m where $m < i$ or $m \geq j$.

The k sets $\{P(n_k, i)\}_{i=0}^{k-1}$ partition the set of all labellings. By construction, the value of $I_\ell(n_k)$ is the same for each label $\mathcal{L}_\ell \in P(n_k, i)$; call this value $V(n_k, i)$. (Here, $V(n_1, 0) = 1$, $V(n_2, 0) = 100$, $V(n_2, 1) = 1$, ..., $V(n_3, 0) = 180$, $V(n_3, 1) = 81$, $V(n_3, 2) = 180$, etc.) Observe that FOCL can compute these values efficiently using a single top-down pass as the values of $V(n_k, i)$ can be computed based on the values of $\{V(n_{k-1}, j)\}_j$. *N.b.,* FOCL *will only deal with these* $V(n_i, j)$ *values; it never needs to explicitly construct the* $P(n_k, j)$ *sets.*

Algorithm FOCL($\mathcal{G}$, $\{d(n_i), s(n_i), u(n_i)\}_i$, R, L, S)

1. For each n_k := ⟨root⟩..⟨leaf⟩
   Compute $\{V(n_k, i)\}_{i=0}^{l-1}$ based on Equation 4, ...
2. For each n_k := ⟨leaf⟩..⟨root⟩
   Compute $\{M(n_k, i)\}_{i=0}^{k-1}$ based on Equation 5, ...
3. Return the optimal labelling, based on the decisions made in determining $M(\langle \mathtt{root} \rangle, 0)$ ...

Figure 4: Code for FOCL

FOCL's second pass [Figure 4's Line 2] works from the bottom up (here from n_4 up to n_1). At each stage, when dealing with n_k, FOCL determines the labellings in each equivalence class that are best from "here down": that is, it computes $M(n_k, i)$, defined to be the smallest value of $E[\mathcal{L}_\ell, n_k]$ over all labelling $\mathcal{L}_\ell \in P(n_k, i)$; *i.e.*,

$$
M(n_k, i) \overset{def}{=} \min\{\ E[\mathcal{L}_\ell, n_k] \mid \mathcal{L}_\ell \in P(n_k, i)\ \} \tag{5}
$$

Working bottom up, FOCL will know the values of $\{M(n_{k+1}, j)\}_j$ when dealing with n_k. It can use the appropriate value from this set in the role of "$E[\mathcal{L}_\ell, n_{k+1}]$", together with the value $V(n_k, i)$ for "$I_\ell(n_k)$" in Equation 3, and then compute the two candidate values for $M[\mathcal{L}_\ell, n_k]$: one based on mapping n_k to C^+, and the other to C^-. FOCL sets $M(n_k, j)$ to be the smaller of these two values. It can then use this information to compute $M(n_{k-1}, i)$, and so on. On reaching the root node, FOCL explicitly has the value of $M(\langle \mathtt{root} \rangle, 0)$, which by construction is the minimal value of $E[\mathcal{L}_\ell] = E[\mathcal{L}_\ell, \langle \mathtt{root} \rangle]$ value over all $\mathcal{L}_\ell \in P(\langle \mathtt{root} \rangle, 0) = \mathcal{LL}(\mathcal{G})$, as desired.

At each stage, FOCL also records whether the preferred label within each $P(n_k, i)$ mapped n_k to C^+ or C^-; it can assemble these mappings to form the optimal labelling. Notice the runtime of the FOCL process is $O(|\mathcal{N}|^2)$.[6]

[6] Given the graph and values shown in Figure 2 and the parameters $R = 2$, $L = 1$ and $S = 10$, the optimal labelling is $\mathcal{L}_{\langle -+-- \rangle}$. Its cost is 29% (resp., 45%) better than the alternative approach of not caching at all (resp., indiscriminately caching everywhere). Section 4 provides a more comprehensive set of examples.

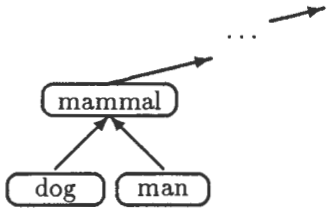


Figure 5: Multiple rules matching a goal

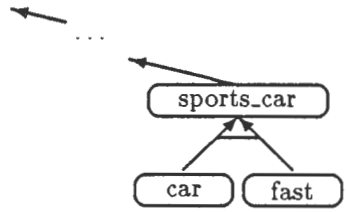


Figure 6: Conjunction in the Inference Graph

3.2 Multiple Rules matching a (Sub)Goal

This subsection deals with the situation where multiple rules can match a goal. Consider Figure 5, taken from the left side of Figure 1's inference graph), and let n_a (resp., n_d, n_m) represent the node whose literal is `mammal(X)` (resp., `dog(X)`, `man(X)`). In this case the nodes in each of the two diverging branches will receive the same number of indirect queries:

$$I_\ell(n_m) = I_\ell(n_d) = \begin{cases} I_\ell(n_a) + d(n_a) & \text{if } \mathcal{L}_\ell[n_a] = \mathrm{C}^- \\ 1 & \text{if } \mathcal{L}_\ell[n_a] = \mathrm{C}^+ \end{cases}$$

As each of these two branches, separately, is a linear knowledge base, we can use the analysis from the previous section. During the upward traversal, notice that Equation 3 continues to hold, even though $Ch(n_a) = \{n_d, n_m\}$ is not a singleton. (As the first term of Equation 3 is a summation over all the node's children, it will incorporate the cost of both the branches.) Thus the cost equation easily generalizes to the case of trees. The other computations remain the same as in the simple linear knowledge base case.

3.3 Conjunctions in Rules

Each rule with a conjunctive antecedent (*i.e.*, each clause with more than one negative literal) corresponds to a more complicated hyper-arc in the inference graph. While computing the cost function for such nodes, we must deal with the extra overhead of finding solutions that satisfy all literals. This process is equivalent to evaluating a join in the relational database [Ull88]. There are various methods of evaluating joins, including selection on an attribute, sort join, multiway merge-sort, join using index, etc. [Ull89]. As sort join seems to work well in general, we will base our discussion on this approach.

To explain the working of the sort join, consider the inference graph shown in Figure 6 (taken from the far right side of Figure 1). To answer the query `sports_car(X)`, we first find all the solutions to `car(X)` and `fast(X)` individually, and then sort each of them independently. Then, in a single traversal of the sorted lists, we find the values that are common to both `car` and `fast`, giving the set of answers to the `sports_car` query.

Now does this affect our formulation? For each query at `sports_car`, there will be one query at each of `car` and `fast`. Thus the expressions for the basic terms remain similar to the previous subsection. The cost for sorting a list of length m is $O(m \log m)$ [AHU87], and of a simple traversal, is $O(m)$. Thus, in the above example if there are m_1 solutions to `fast` and m_2 solutions to `car`, the asymptotic cost of evaluating the conjunction is bounded by $K(m_1 \log m_1 + m_2 \log m_2 + m_1 + m_2)$ where K is a constant dependent on the implementation. This value needs to be added to the cost expression Equation 3.

When there are more than two subgoals in a conjunction, we need to use multi-way joins [Ull89]. The basic process, however, remains the same.

3.4 Computing the Parameters

In any given environment, these values of the three implementation dependent parameters, L, R and S, are standard and should be known from the supplier's data. The three functions, $u(n_k)$, $d(n_k)$ and $s(n_k)$, are specific to an application and have to be obtained by the user of the FOCL algorithm.

Fortunately, we can estimate these values based on the statistics that are maintained by several commercial database systems [SAC+79]. We can periodically collect these statistics (*e.g.*, each time the system is "re-compiled"), and use these values as (estimates of) $s(n_k)$, $u(n_k)$ and $d(n_k)$ when computing the appropriate caching label. We can also use statistical measures to bound our confidence in these estimates; see [Gre92].

Notice that both $d(n_i)$ and $u(n_i)$ are with respect to some interval of time — either the "lifetime" of the overall system, or the time during which this caching strategy is "in effect", which can be the interval of time between a pair of re-compilations.

4 Empirical Results

This section compares the performance of three IMS systems: IMS_{opt} that uses the *optimal caching scheme* obtained by FOCL, $\mathrm{IMS}_\forall$ that uses the *cache everywhere scheme* of [SM91], and $\mathrm{IMS}_\neg$ that uses the *no cache scheme*.

Experimental Setup: We determined the values of $E[\mathrm{IMS}_{opt}]$, $E[\mathrm{IMS}_\forall]$ and $E[\mathrm{IMS}_\neg]$ in 54 different contexts. Each context is defined in terms of a particular knowledge base and specific distribution of queries, specified below. All simulations use the same cost parameters $R = 2$, $L = 1$ and $S = 10$, obtained from experimental data [Deb90]. The depth of each knowledge base is set at 6, which is considered typical for real applications. Furthermore, the only atomic database facts that match a node are at the leaves; *i.e.*, $s(n_i) = 0$ for all internal nodes n_i. We also specify that each node $n \in \mathcal{N}$ is updated at the rate of $u(n) = 0.01$, *i.e.*, once every one hundred queries.

We considered three clusters of experiments, which differ in terms of the number of ground instances that match each leaf predicate. Figure 7 (resp., Figure 8, Fig-

ure 9) describes 18 contexts in which each leaf predicate matches exactly 1 (resp., 5, 10) database literals.

Each cluster of $18 = 6 \times 3$ contexts is formed as the cross-product of 6 different knowledge bases, depending on whether the branching factor from goal to subgoals was 1..6,[7] and three different query distributions: In distribution `Dist1`, the root receives 10 queries (*i.e.*, $d(\langle \texttt{root} \rangle) = 10$), each node at the next level receives 20, then 30 for each node at level 3, etc. In `Dist2`, this is reversed — each leaf node receives 10 queries, and the root, 60, as there are 6 levels. In `Dist3`, every node at every level receives same number of queries. ([CG92] specifies this data more precisely.)

Each graph plots $\frac{E[\text{IMS}_\forall] - E[\text{IMS}_{opt}]}{E[\text{IMS}_{opt}]}$ and $\frac{E[\text{IMS}_\neg] - E[\text{IMS}_{opt}]}{E[\text{IMS}_{opt}]}$, versus the branching factor, for each of its 18 contexts. Notice that better IMS systems have *smaller* $\frac{E[\text{IMS}] - E[\text{IMS}_{opt}]}{E[\text{IMS}_{opt}]}$ values. As the optimal IMS_{opt} system has the uniform value 0, we did not plot values for the IMS_{opt} system in these graphs.

Experiment Cluster 1: 1 Ground Instance per Leaf node: These results appear in Figure 7. For small values of ground instances (*i.e.*, for branching factor $\approx$ 1), the best scheme is to cache everywhere and therefore the cost obtained by $\text{IMS}_\forall$ is the same as the cost of IMS_{opt}. This does *not* hold for larger branching factors, however; for branching factor is 6, IMS_{opt}'s cost is 55% better than $\text{IMS}_\forall$. The opposite is true for $\text{IMS}_\neg$ as the degradation in cost is more pronounced for low values of the branching factor. (It is 700% when the branching factor is 1. This actual value is too high to be shown in Figure 7.) While the actual improvements vary with the specific query distribution, the improvements are significant in all cases — an average of 30%. However, it seems that `Dist2`, with more queries at "higher" nodes, favours $\text{IMS}_\forall$, which makes sense as there can be more saving in the inference cost, regardless of the query distribution.

Experiment Cluster 2: 5 Ground Instances per Leaf node: As shown in Figure 8, $\text{IMS}_\forall$'s cost is far worse than IMS_{opt}'s in most of these cases. In fact, $\text{IMS}_\neg$ is closer to optimum, even though it can be worse by as much as 50%. $\text{IMS}_\forall$ performs well only when the size of knowledge base is small (five rules, one ground instance) or when the query distribution is uniform (`Dist3`). Since $\text{IMS}_\forall$ caches everywhere, it fails to respond to the variations in query distribution. This effect was less pronounced when the number of instances was lower, as in the previous experiment.

Experiment Cluster 3: 10 Ground Instances per Leaf node: As shown in Figure 9, $\text{IMS}_\neg$ seems to be

[7] Hence, the knowledge bases ranged from 5 rules and 1 ground instance (in Figure 7's framework, when the branching factor is 1) to about 10,000 rules and approximately 100,000 ground instances (in Figure 9's framework, when the branching factor is 6). Note that these are precisely the kind of knowledge bases envisaged to be required in the future applications [MCPT91].

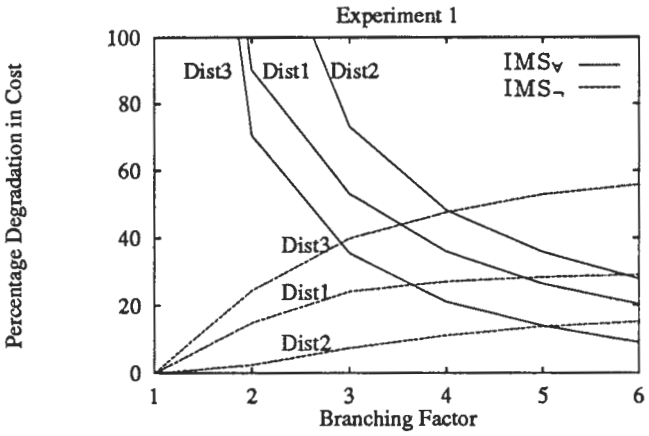


Figure 7: Results for Experiment Cluster 1

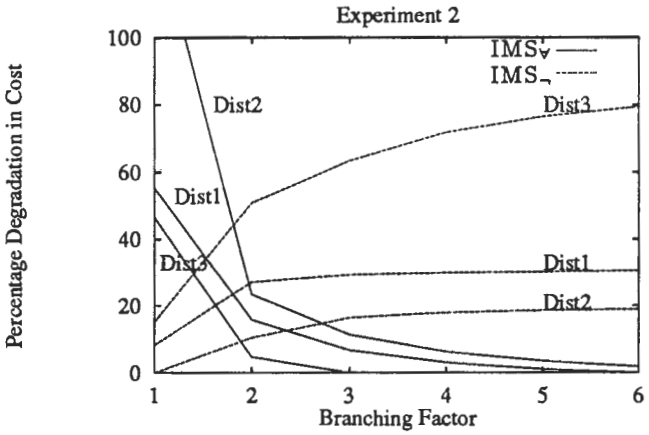


Figure 8: Results for Experiment Cluster 2

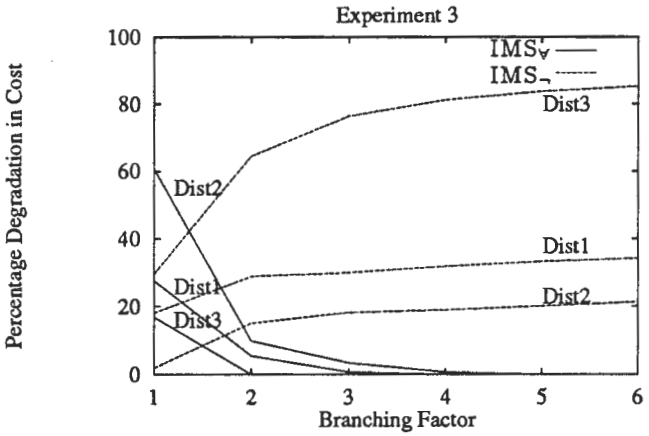


Figure 9: Results for Experiment Cluster 3

almost optimal here; in most cases within 10%. $IMS_{\forall}$ does worse in general, especially with larger branching factors.

Summary of Experiments: These experiments suggest that $IMS_{\forall}$.is appropriate only for small number of ground instances; this explains the results obtained by [SM91]. However, this approach does not hold for larger knowledge bases. Given a large number of ground instances, it may be better to use $IMS_{\neg}$ rather than $IMS_{\forall}$. Of course, neither of these can be better than IMS_{opt}, which is guaranteed to have the best performance in all cases.

5 Conclusions

This work can be extended in a few directions: To deal with inference graphs that are not tree-shaped (*e.g.*, redundant [Gre91], recursive [SGG86], etc.); to estimate the number of solutions that can be cached at each node without actually running the inference process; to use sampling to approximate the distribution of queries and updates [LN90, Gre92]; and to deal with different cost models — *e.g.*, to include the storage cost of maintaining cached values, or to allow the values of the various parameters (*e.g.*, R, L, S) to vary with the size of knowledge base, or the number of variables involved, etc.

To recap: this paper first presents a formal definition of caching and a quantitative model that formalizes the interactions among the various parameters that affect caching performance. We use this model to design FOCL, an efficient algorithm that computes the optimal cache labelling for any "tree shaped" knowledge base; *i.e.*, FOCL determines which literals should cache their solutions to obtain an IMS system whose overall cost (for answering a given distribution of queries, given a specific distribution of updates, etc.) is minimal. The paper also presents a set of experiments to illustrate that the FOCL-based IMS_{opt} can outperform systems based on either of the standard caching strategies (no caching or universal caching), across a broad range of contexts.

References

[AHU87] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structure and Algorithms.* Addison-Wesley Publishing Company, 1987.

[CG92] Vinay K. Chaudhri and Russell Greiner. A formal analyis of caching in backward chaining systems. Technical Report forthcoming, University of Toronto, 1992.

[CM81] William F. Clocksin and Christopher S. Mellish. *Programming in Prolog.* Springer-Verlag, New York, 1981.

[Deb90] S. K. Debray. Personal Communication, 1990.

[Gre91] Russell Greiner. Finding the optimal derivation strategy in a redundant knowledge base. *Artificial Intelligence*, 50(1):95–116, 1991.

[Gre92] Russell Greiner. Learning efficient query processing strategies. In *Proceedings of Eleventh ACM SIGACT/SIGMOD Symposium on Principles of Database Systems*, 1992.

[LN90] R.J. Lipton and J.F. Naughton. Query size estimation by adaptive sampling. In *Proceedings of Ninth ACM SIGACT/SIGMOD Symposium on Principles of Database Systems*, pages 40–46, 1990.

[MCPT91] John Mylopoulos, Vinay K. Chaudhri, Dimitris Plexousakis, and Thodoros Topaloglou. Four Obvious Steps Towards Knowledge Base Management. In *Proceedings of the Second International Workshop on Intelligent and Cooperative Information Systems: Core Technology for Next Generation Information Systems*, pages 28–30, Como, Italy, October 1991.

[Mos83] M. G. Moser. An Overview of NIKL, the new implementation of KL–ONE. In C. L. Sidner, editor, *Research in Knowledge Representation and Natural Language Understanding – Annual Report, 1 September 1982 - 31 August 1983*, pages 7–26. Bolt Beranek and Newman, 1983.

[Nem66] George L. Nemhauser. *Introduction to Dynamic Programming.* John Wiley and Sons, Inc., New York, 1966.

[SAC+79] P.G. Selinger, M.M. Astrahan, D.D. Chamberlin, R.A. Lorie, and T.G. Price. Access Path Selection in a Relational DBMS. In *Proceedings of the 1979 SIGMOD Conference*, 1979.

[Sel89] Timos K. Sellis. Efficiently supporting procedures in a relational database system. In *Proceedings of the 1987 SIGMOD Conference*, Detroit, August 1989.

[SGG86] D. E. Smith, M. R. Genesereth, and M. L. Ginsberg. Controlling recursive inference. *Artificial Intelligence*, 30(3):343–89, 1986.

[SJGP90] Michael Stonebraker, Anant Jhingaran, Jeffrey Goh, and Spyros Potamianos. On rules, procedures, caching and views in databases. Technical Report UCB/ERL M90/36, University of California, Berkeley, April 1990.

[SM91] G. Schmolze and William S. Mark. The NIKL Experience. *Cognitive Science*, 6(2):48–69, February 1991.

[Ull88] Jeffrey Ullman. *Principles of Data Base and Knowledge Base Systems*, volume 1. Addison Wesley, 1988.

[Ull89] Jeffrey Ullman. *Principles of Data Base and Knowledge Base Systems*, volume 2. Addison Wesley, 1989.

Reasoning about Action in First-Order Logic*

Charles Elkan
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, California 92093-0114

ABSTRACT: The main contribution of this paper is a method of axiomatizing actions and their effects in standard first-order logic in a way that solves the frame problem. It has long been thought that the frame problem makes monotonic logics inadequate for reasoning about action; this paper shows how to achieve the benefits of a nonmonotonic logic using a monotonic logic. The use of first-order logic permits considerable simplicity and a frame axiom that is explicit, fixed, and intuitive. Moreover, complex worlds of actions can be axiomatized: fluents can be momentary or dynamic, the preconditions of actions need not be fully known, and the ramifications of actions can be stated unambiguously.

1 Introduction

This paper discusses reasoning about actions and their effects using the ontology of states, actions, and fluents, known as the situation calculus. The main positive contribution of the paper is a method of axiomatizing actions in standard first-order logic that solves the frame problem. It has long been thought that this problem makes monotonic logics inadequate for reasoning about action. This paper shows that the introduction of a *cancels* predicate, in addition to the more usual *holds* and *causes* predicate, and the use of bidirectional implication as a minimization operator, effectively allows nonmonotonic reasoning about actions to be performed using first-order logic.

Section 2 below introduces the situation calculus and the frame problem using the Yale shooting problem as an example. Section 3 discusses criteria for evaluating candidate solutions to the frame problem, and Section 4 then presents our solution, which satisfies the criteria advanced previously. Section 5 explains extensions to the method to accommodate transient properties of the world, to solve the (representational) qualification problem, and to solve the ramification problem: how to state the indirect effects of actions in a modular fashion. Finally, Section 6 compares the methods of this paper to related work.

*The preparation of this paper was supported by a grant from the Powell Foundation and by Grant No. IRI-9110813 from the National Science Foundation.

2 The situation calculus and the frame problem

The most complex theories of action that have been treated formally in AI research involve multiple agents with differing beliefs and agents. in some of these theories the temporal properties of actions are also considered: they take time, they may be concurrent or overlap. Substantial epistemic and temporal reasoning requires a logic with a theory of time and knowledge built into its structure [Moore, 1985; Konolige, 1986; Cohen *et al.*, 1990]. Solving the frame problem inside such logics, which are typically modal, is beyond the scope of this paper.

This work reported here is directed towards reasoning about the actions of a single agent. At this level of detail, the traditional ontology is called the situation calculus [McCarthy and Hayes, 1969]. The categories of this ontology are *states*, *objects*, *fluents*, and *actions*. Briefly, a *state* is a snapshot of the world, an *object* is something whose identity is preserved across states, a *fluent* is a relationship between objects that holds in some states, and an *action* is an event that changes one state into another.

States, objects, fluents, and actions can all be represented as terms in a first-order language. For example, if a and b are constants designating two child's building blocks, then $stack(a, b)$ can designate the action of placing a on b, where $stack(\cdot, \cdot)$ is a function symbol. Further, if s_0 is a constant designating an an initial state, then $do(s_0, stack(a, b))$ designates the state resulting from performing the action $stack(a, b)$.[1] An example of a term representing a fluent is $on(a, b)$, which presumably means that block a is on top of block b. The standard approach to encoding the situation calculus in first-order logic uses a special *holds* predicate. The arguments of *holds* are a fluent and a state, as in $holds(on(a, b), do(s_0, stack(a, b)))$.

[1] The function symbol $do(\cdot, \cdot)$ is sometimes named $result(\cdot, \cdot)$. Contrary to the usual convention, the first argument of $do(\cdot, \cdot)$ is a state, and the second argument is an action. This convention allows the state resulting from a sequence of actions $a_1, a_2, \ldots a_k$ to be written perspicuously as $do(\cdots do(do(s_0, a_1), a_2), \cdots, a_k)$.

In the world of the Yale shooting problem [Hanks and McDermott, 1986][2] there are two objects, a person Fred and a gun, three fluents, *loaded*, *alive*, and *dead*, and three actions, *load*, *wait*, and *shoot*. The fluents and actions have their obvious meanings and the person and the gun are left unnamed because they are fixed.

Only the *load* and *shoot* actions have significant effects. Using the *holds* predicate these can be specified by the following axioms:

$$\begin{aligned}
&\forall s\ holds(loaded, do(s, load))\\
&\forall s\ holds(loaded, s) \rightarrow holds(dead, do(s, shoot))\\
&\forall s\ holds(loaded, s) \rightarrow \neg holds(alive, do(s, shoot))\\
&\forall s\ holds(loaded, s) \rightarrow \neg holds(loaded, do(s, shoot)).
\end{aligned}$$

It should be possible to solve inference problems of various types using an axiomatization of a world of actions and their effects. Two general classes of problems are those involving reasoning forward in time, and those requiring reasoning backwards. These can be called *projection* and *retrojection* problems respectively. More precisely, a projection problem is one where it is known what holds in an initial state, and the task is to draw conclusions about what holds after a certain sequence of actions. Retrojection problems are also loosely called explanation problems [Hanks and McDermott, 1987]. They require inferring what held before a given sequence of actions, given observations of what holds afterwards.

The original paper on the Yale shooting world discussed only a projection problem: given that Fred is alive to start with, is he dead in the state after loading the gun, then waiting, then shooting? Formally, given

$$holds(alive, s_0).$$

the task is to prove

$$holds(dead, do(do(do(s_0, load), wait), shoot)).$$

One's innocent hope is that the four axioms above about the effects of the *shoot* and *load* actions are sufficient. Unfortunately, simply stating the effects of actions is not enough. To reach the desired conclusion at least the extra axiom

$$\forall s\ holds(loaded, s) \rightarrow holds(loaded, do(s, wait)).$$

is needed. To answer other projection and retrojection questions, many further axioms are necessary:

$$\begin{aligned}
&\forall s\ holds(alive, s) \rightarrow holds(alive, do(s, load))\\
&\forall s\ holds(alive, s) \rightarrow holds(alive, do(s, wait))\\
&\forall s\ holds(alive, s) \wedge \neg holds(loaded, s)\\
&\qquad\qquad \rightarrow holds(alive, do(s, shoot))
\end{aligned}$$

and so on.

In general, if the world consists of a different actions and e potential effects (that is, fluents), it seems that on the order of ae axioms are needed. Almost all these axioms seem redundant: they express the typical case where an action does not influence a property of the world. These axioms, of the form

$$\forall s\ holds(p, s) \wedge \Pi(s) \rightarrow holds(p, do(s, a)),$$

where $\Pi(s)$ is a conjunction of *holds* conditions all referring to the same state s, are called frame axioms. The frame problem is to escape the predicament of having to state so many axioms that should somehow be true by default.

3 Judging candidate solutions to the frame problem

The general issue behind the frame problem is to find a good way to state the relationships between actions and fluents in the situation calculus ontology. Papers on the frame problem generally propose a new logic and present examples of using that logic to express actions and effects. Most papers, however, do not discuss how to evaluate the candidate solutions to the frame problem that they advance. This section identifies three dimensions of adequacy along which these proposed solutions can be judged. For a more detailed and general discussion of these criteria, see [Elkan, 1991].

To constitute a successful general solution of the problem of axiomatizing worlds of actions and their effects, a logic and a methodology for using the logic should give rise to axiomatizations that are declarative, parsimonious, and localist. These criteria seem more or less independent and exhaustive, but we shall not defend that claim. Rather, the remainder of this section is an attempt to give each of the criteria a definition that can be tested as objectively as possible.

Declarative. A declarative method of axiomatizing actions is one that uses a logic with a well-defined semantics specified independently of the rules of inference of the logic. Methods for representing actions that rely on a procedural definition of entailment, such as the STRIPS formalism and its variants [Lifschitz, 1986], do not satisfy this criterion. The main reason that a well-defined semantics is desirable is that it allows one to obtain high confidence in the correctness of a set of axioms, by working out its consequences in two independent ways, model-theoretically and proof-theoretically, and checking that these correspond.

Parsimonious. Parsimony is relatively easy to define. Suppose there are a different relevant actions, and e different relevant effects. An axiomatization that uses $O(ae)$ space is not parsimonious. If each action influences at most d properties of the world, then a parsimonious axiomatization should use at most $O(ad)$ space.[3]

Localist. Ideally, each action-effect combination should be specified by a separate axiom, or by no axiom at all if it is a special case of some general frame axiom. However, it is somewhat arbitrary what is considered a separate axiom. In classical logics, two assertions can be viewed as separate axioms if the logical connective that

[2] It is worth stressing that the contribution of this paper is not yet another "solution" to the Yale shooting problem. The contribution is a general method for axiomatizing the effects of actions. The Yale shooting problem is used as an expository example simply because it is well-known. Note also that there is no attempt here to state all claims with the maximum degree of generality.

[3] A criterion of parsimony is implicit in [Shoham, 1986].

links them implicitly is 'and'; one can imagine privileging a different connective. Moreover, nonclassical logics often have a semantics that is entirely formal, but not compositional: the meaning of a set of axioms cannot be defined in terms of independent meanings for each axiom in the set. Any type of closed world assumption, for example, destroys compositionality. Without compositionality it is not clear why separate axioms are desirable. In addition, there is no necessary correlation between separate axioms and efficient inference. Many inference procedures preprocess input sets of axioms, and represent seemingly modular sets of axioms in unmodular ways. In view of the considerations just outlined, we shall admit as "localist" any axiomatization that only has to be changed in a fixed, small number of places when a new action is taken into account.

4 The general method of axiomatizing actions

Our approach to solving the frame problem uses three basic predicates: *holds*, *causes*, and *cancels*. The arguments of *holds* are a fluent and a state, as in $holds(on(a,b), do(s_0, stack(a,b)))$. The arguments of *causes* and *cancels* are an action, a state, and a fluent, as in

$$causes(stack(a,b), s_0, on(a,b)).$$

The single, fixed, general frame axiom is

$$\forall a,s,p \quad holds(p, do(s,a)) \leftrightarrow \quad (1)$$
$$causes(a,s,p) \vee (holds(p,s) \wedge \neg cancels(a,s,p)).$$

This axiom has a common sense interpretation. It states that a fluent holds in the state resulting from an action if and only if the action "causes" the fluent, or the fluent held before the action, and the action does not "cancel" it. Note that both the *causes* and *cancels* predicates have state arguments, so whether or not an action influences a fluent can depend on the state in which the action is taken.

Using the three basic predicates, the relationships of these fluents and actions can be specified by the following axioms:[4]

$$\forall a,s,p \quad causes(a,s,p) \leftrightarrow \quad (2)$$
$$(a = load \wedge p = loaded) \vee$$
$$(a = shoot \wedge p = dead \wedge holds(loaded, s))$$
$$\forall a,s,p \quad cancels(a,s,p) \leftrightarrow \quad (3)$$
$$(a = shoot \wedge p = alive \wedge holds(loaded, s)) \vee$$
$$(a = shoot \wedge p = loaded \wedge holds(loaded, s)).$$

Axioms (1)–(3) describe the Yale shooting world. They can be used to solve many different inference problems, always using the standard semantics of first-order predicate calculus. For example, in all models of sentences (1)–(3) and $holds(alive, s_0)$, it is the case that

$$holds(dead, do(do(do(s_0, load), wait), shoot))$$

is true. The shooting problem is thus solved.

The so-called murder mystery is a retrojection problem, introduced in [Baker, 1989], The scenario is that Fred is alive initially but not after shooting the gun and then waiting:

$$holds(alive, s_0) \wedge$$
$$\neg holds(alive, do(do(s_0, shoot), wait)).$$

The question is to discover when Fred died, and whether the gun was initially loaded. Axioms (1)–(3) and the sentence immediately above entail

$$holds(loaded, s_0) \wedge cancels(shoot, s_0, alive).$$

The mystery is solved.

A natural question is whether all axiomatizations of the type described above allow all projection problems to be solved. The following theorem states this question precisely, and affirms that the answer is yes.

Theorem: Let the theory T consist of the frame axiom (1), a *causes* axiom of the form

$$\forall a,s,p \quad causes(a,s,p) \leftrightarrow$$
$$\vdots$$
$$a = \alpha \wedge p = \pi \wedge \Pi_{\alpha,\pi}$$
$$\vdots$$

a *cancels* axiom of the form

$$\forall a,s,p \quad cancels(a,s,p) \leftrightarrow$$
$$\vdots$$
$$a = \alpha \wedge p = \pi \wedge \Omega_{\alpha,\pi}$$
$$\vdots$$

and an initial state axiom of the form

$$\forall p \quad holds(p, s_0) \leftrightarrow$$
$$\ldots \vee p = \pi_k \vee \ldots$$

In these axioms each subformula $\Pi_{\alpha,\pi}$ and $\Omega_{\alpha,\pi}$ is a conjunction of preconditions that may involve local existential quantification. That is, each $\Pi_{\alpha,\pi}$ and $\Omega_{\alpha,\pi}$ is of the form

$$\exists \bar{x} \quad holds(f_1(\bar{x}), s) \wedge \cdots holds(f_k(\bar{x}), s)$$

where each f_i is a fluent and $\bar{x}$ allows fluents to be parametrized. Then T is consistent, and for all fluents p and action sequences $a_1, a_2, \ldots a_k$, either

$$T \models holds(p, do(\cdots do(do(s_0, a_1), a_2), \cdots, a_k))$$

or

$$T \models \neg holds(p, do(\cdots do(do(s_0, a_1), a_2), \cdots, a_k)).$$

Proof: Induction over the length of the action sequence $a_1, a_2, \ldots a_k$. ■

A planning problem is a projection problem where the action sequence is unknown. Given a goal fluent p, the

[4] Almost every paper on the Yale shooting problem uses a slightly different set of fluents. The three used here are from the original circumscriptive attempt to solve the problem [Hanks and McDermott, 1986]. Our solution does not depend on this choice of fluents.

task is to find an action sequence $\alpha_1, \alpha_2, \ldots \alpha_k$ for some k such that

$$T \models holds(p, do(\cdots do(do(s_0, \alpha_1), \alpha_2), \cdots, \alpha_k)).$$

A corollary of the theorem above is that planning is semi-decidable: for any set of goal fluents, if a plan exists that achieves them simultaneously, it can be found. Planning should in fact be efficiently decidable using axioms of the form proposed here, since these axioms correspond to the completion [Clark, 1978] of logic program clauses. For the particular axioms given as an example above, the corresponding logic program is

$$causes(A, S, P) \rightarrow holds(P, do(S, A))$$
$$holds(P, S) \wedge \neg cancels(A, S, P) \rightarrow holds(P, do(S, A))$$
$$causes(load, S, loaded)$$
$$holds(loaded, S) \rightarrow causes(shoot, S, dead)$$
$$holds(loaded, S) \rightarrow cancels(shoot, S, alive)$$
$$holds(loaded, S) \rightarrow cancels(shoot, S, loaded).$$

Projection and planning problems can be reduced to query-answering against this pure PROLOG program. The state of the art in PROLOG implementation is about nine RISC cycles per logical inference [Mills, 1989]. One could in principle obtain a specialized reasoner running at a comparable speed for any world of actions and fluents axiomatized following our guidelines.

5 Solving further problems in axiomatizing actions

Incomplete knowledge. The qualification problem is that any formalization of a fragment of common sense knowledge is always incomplete. However many preconditions have been taken into account in the formal statement of a rule, further preconditions exist whose truth influences the truth of the consequent of the rule, so the formal rule should have even more antecedents. Given that one can never be sure that all relevant preconditions for an action have been mentioned, the issue from a knowledge representation perspective is to axiomatize actions and their effects in such a way that when new preconditions are discovered, they can be stated easily [McCarthy, 1977]. For example in the Yale shooting world, one should be able to add the condition that the gun cannot be loaded if it is locked away.

In our framework, the preconditions of actions can be stated in a modular way using a new predicate *prevented*. Here is an example:

$$\forall a, s, p \quad causes(a, s, p) \leftrightarrow \neg prevented(a, s) \wedge$$
$$\vdots$$
$$(a = load \wedge p = loaded) \vee$$
$$\vdots$$
$$\forall a, s \quad prevented(a, s) \leftrightarrow$$
$$\vdots$$
$$(a = load \wedge holds(lockedaway, s))$$
$$\vdots$$

When a new precondition for an action is discovered, the use of the *prevented* predicate allows it to be taken into account by changing an axiomatization in just one place.

Using a *prevented* predicate in the way just described allows an axiomatization that is discovered to be incomplete to be updated in a straightforward way. A second issue of incomplete knowledge is how to assert in an axiomatization that the preconditions of an action are only partially stated. Intuitively, the $\leftrightarrow$ connective in the axiom defining *prevented* says that *prevented* is fully defined: there are no a and s such that the truth or falsity of $prevented(a, s)$ is unknown. Conditionally replacing the $\leftrightarrow$ connective with a unidirectional $\leftarrow$ connective permits the truth of $prevented(a, s)$ to be partially unspecified. The following axioms say that $prevented(a, s)$ is true if $prevents(a, s)$ is true, and if $determined(a, s)$ is true, then $prevented(a, s)$ is true only if $prevents(a, s)$ is true. That is, *prevented* is fully specified for those a and s for which *determined* is true, and partially specified for all other a and s.

$$\forall a, s \quad prevented(a, s) \leftarrow prevents(a, s)$$
$$\forall a, s \quad (prevented(a, s) \rightarrow prevents(a, s)) \leftarrow determined(a, s))$$
$$\forall a, s \quad prevents(a, s) \leftrightarrow$$
$$\vdots$$
$$(a = load \wedge holds(lockedaway, s))$$
$$\vdots$$
$$\forall a, s \quad determined(a, s) \leftrightarrow$$
$$\vdots$$
$$a = load$$
$$\vdots$$

Momentary and dynamic fluents. Once they start to hold, most properties of the world tend to continue to hold. The frame axiom (1) formalizes this observation. However some properties of the world tend to change by themselves. Fluents of this nature are called *momentary* [Lifschitz and Rabinov, 1989], and a flexible logic for reasoning about action must allow them to be decoupled from any frame axioms. The following modified axiom achieves this in our approach:

$$\forall a, s, p \quad holds(p, do(s, a)) \leftrightarrow$$
$$causes(a, s, p) \vee$$
$$(holds(p, s) \wedge \neg momentary(p) \wedge \neg cancels(a, s, p)).$$

This axiom is identical to (1), except a fluent p holding in a state s may cease to hold because it is momentary as well as because it is canceled. For an example of how to use this axiom, consider the loud noise presumably caused by a *shoot* action. This fluent can be named *bang* and axiomatized by adding a description of what induces it to the *causes* axiom:

$$\forall a, s, p \quad causes(a, s, p) \leftrightarrow$$
$$\vdots$$

$$(a = shoot \wedge p = bang \wedge holds(loaded, s)) \vee$$
$$\vdots$$

and stating that it is momentary:

$$\forall p \ \ momentary(p) \leftrightarrow p = bang.$$

A dynamic fluent is a property of the world whose nature is to change spontaneously [Lifschitz and Rabinov, 1989]. For example, the level of water in a swimming pool that is being filled changes even after unrelated actions. Dynamic properties of the world are difficult to axiomatize in any logic that uses the situation calculus ontology, because they are typically time-dependent, and time is deliberately excluded from the situation calculus, in a tradeoff of tractability for expressiveness [Levesque, 1988]. However if one assumes that all actions take the same amount of time, for example, then dynamic fluents can be accommodated. One can write

$$\forall a, s, p \ \ causes(a, s, p) \leftrightarrow$$
$$\vdots$$
$$\exists k \ p = level(pool, k) \wedge$$
$$holds(level(pool, k-1), s) \wedge holds(filling(pool), s)$$
$$\vdots$$
$$\forall a, s, p \ \ cancels(a, s, p) \leftrightarrow$$
$$\vdots$$
$$\exists k \ p = level(pool, k) \wedge$$
$$holds(level(pool, k), s) \wedge holds(filling(pool), s)$$
$$\vdots$$

The ramification problem. This problem is to axiomatize actions in such a way that their indirect effects can be stated independently. For example, suppose that the world of interest is a living room with two air vents. If both vents are blocked the room will be stuffy. One does not want to state this repeatedly as a potential effect of every action that may block a vent. The standard solution in the literature [Ginsberg and Smith, 1988; Baker and Ginsberg, 1989] to the ramification problem is to use so-called domain constraints that state relationships between fluents without mentioning actions at all. Following this approach one would write

$$holds(stuffy, s) \equiv \tag{4}$$
$$holds(blocked(vent_1), s) \wedge holds(blocked(vent_2), s).$$

Axioms of this sort are acausal: they are silent as to what events in the world have the power actually to induce other events. Unfortunately they often lead to unwanted ambiguity. Suppose that

$$holds(blocked(vent_1), s_0) \wedge \neg holds(blocked(vent_2), s_0)$$

is true. This entails $\neg holds(stuffy, s_0)$. What can one conclude is true after an action a given the fact $causes(a, s_0, blocked(vent_2))$? All that can be deduced using (4) is

$$cancels(a, s_0, blocked(vent_1)) \vee causes(a, s_0, stuffy).$$

Intuitively, there should be no ambiguity: insufficient ventilation cannot normally cause a vent to become unblocked.. We want to conclude

$$causes(a, s_0, stuffy).$$

This ambiguity problem is discussed in [Ginsberg and Smith, 1988], but no satisfactory solution is presented. The ambiguity can be resolved in our approach, because we have a theory of causation that, however limited, is explicit. We can state that if certain fluents are caused or canceled, then other fluents are also caused or canceled, given perhaps that various preconditions hold. In the present example, we can write

$$\forall a, s, p \ \ causes(a, s, p) \leftrightarrow$$
$$\vdots$$
$$(p = stuffy \wedge holds(blocked(vent_1), s)$$
$$\wedge \ causes(a, s, blocked(vent_2))) \vee$$
$$(p = stuffy \wedge holds(blocked(vent_2), s)$$
$$\wedge \ causes(a, s, blocked(vent_1))) \vee$$
$$\vdots$$

This axiom captures reality better: it entails $causes(a, s_0, stuffy)$ as wanted.

The real issue is that if the vents are blocked, that brings about stuffiness *causally*, whereas in the other direction, stuffiness only suggests that the vents are blocked *evidentially*. Causal and evidential implications have different logical behaviours [Pearl, 1988]. It does not work to use the same $\rightarrow$ connective to encode both, whether the connective is understood inside a monotonic or a nonmonotonic logic.

One could still object that stating causal relationships explicitly is less parsimonious. However this difference is unimportant: computationally an undirected axiom such as (6) has to be indexed to make it usable whenever an instance of any of its left- or right-hand-side conjuncts becomes true or false. If anything, inference is more efficient with a causal description of indirect effects.

6 Discussion

The simplicity of the ideas for reasoning about action introduced in this paper should not be misconstrued. Basing a solution to the frame problem on a weak, but explicit, theory of causality has been suggested before, notably in [Lifschitz, 1987]. The general idea is to specify what actions induce what effects, and to use the minimization machinery of a nonmonotonic logic to say that actions induce no other effects. There are two principal new ideas here: the introduction of a *cancels* predicate dual to the *causes* predicate, and the observation that bidirectional implication is a powerful minimization operator that can be used to obtain nonmonotonic effects.

Given any *fixed* information about a world of actions and their effects, all valid conclusions about that world can be obtained by monotonic inference using first-order logic. If *new* information about an action or an effect must be taken into account, then the truth of some conclusions may change. New information can therefore not

be taken into account just by adding new axioms to the prior axiomatization. However, with an axiomatization constructed according to the method laid out above, new information can be taken into account by modifying the prior axiomatization in a straightforward, precisely specified, computationally easy way. Some nonmonotonic logics appear to allow new information to be taken into account just by adding new axioms, but in those logics, after new axioms are added, the overall minimization called for in the semantics of the logic must be performed again. This process is analogous to inserting new disjuncts in the right hand side of an $\leftrightarrow$ sentence, which is how new facts are included in the axiomatizations advocated in this paper.

An alternative approach to axiomatizing the effects of actions in first-order logic has recently been published by Reiter [Reiter, 1991], building on work by Pednault [Pednault, 1989] and Schubert [Schubert, 1989]. The methods of this paper are considerably simpler, for two main reasons. First, Reiter uses multiple frame axioms, each one involving quantification over actions and states. A single frame axiom is sufficient above, because it involves quantification over fluents in addition. Second, no distinction is made in this paper between the preconditions under which an action is possible and the preconditions which must hold for it to have a particular effect. The absence of this distinction makes the theorem of Section 4 much simpler than the corresponding theorem in [Reiter, 1991].

Some further contrasts between the techniques of this paper and alternative approaches to formalizing the effects of actions are worth mentioning. The TWEAK calculus of [Chapman, 1987] is inherently incapable of solving the ramification problem. Approaches based on a separate logical theory or model for each state of the world [Ginsberg and Smith, 1988; Rao and Foo, 1989], with actions corresponding to database updates, do not have clear frame axioms. Approaches based on circumscription [Lifschitz and Rabinov, 1989; Baker and Ginsberg, 1989] do not have practical proof procedures.

Acknowledgement: Many of the ideas here first arose in discussions with Ray Reiter.

References

[Baker and Ginsberg, 1989]
Andrew B. Baker and Matthew L. Ginsberg. Temporal projection and explanation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 906–911, August 1989.

[Baker, 1989] Andrew B. Baker. A simple solution to the Yale shooting problem. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 11–20, 1989.

[Chapman, 1987] David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.

[Clark, 1978] Kenneth. L. Clark. Negation as failure. In Hervé Gallaire and Jack Minker, editors, *Logic and Databases*, pages 293–322. Plenum Press, New York, 1978.

[Cohen *et al.*, 1990] Philip R. Cohen, Hector J. Levesque, and Joseph Nunes. On acting together: Joint intentions for intelligent agents. In *Proceedings of the National Conference on Artificial Intelligence*, Boston, Massachusetts, August 1990.

[Elkan, 1991] Charles Elkan. Formalizing causation in first-order logic: Lessons from an example. In *Working Notes of the AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, Stanford University, March 1991.

[Ginsberg and Smith, 1988] Matthew L. Ginsberg and David E. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence*, 35(2):165–195, June 1988.

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

[Hanks and McDermott, 1987] Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379–412, November 1987.

[Konolige, 1986] Kurt Konolige. *A Deduction Model of Belief.* Pitman, 1986.

[Levesque, 1988] Hector J. Levesque. Logic and the complexity of reasoning. *The Journal of Philosophical Logic*, 17:355–389, 1988.

[Lifschitz and Rabinov, 1989] Vladimir Lifschitz and Arkady Rabinov. Things that change by themselves. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 864–867, August 1989.

[Lifschitz, 1986] Vladimir Lifschitz. On the semantics of STRIPS. In *Proceedings of the Workshop on Reasoning about Actions and Plans*, pages 1–9. Morgan Kaufmann Publishers, Inc., 1986.

[Lifschitz, 1987] Vladimir Lifschitz. Formal theories of action. In *Proceedings of the Workshop on the Frame Problem in Artificial Intelligence*, Lawrence, Kansas, 1987.

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

[McCarthy, 1977] John McCarthy. Epistemological problems of artificial intelligence. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 1038–1044, 1977.

[Mills, 1989] Jonathan W. Mills. A pipelined architecture for logic programming with a complex but single-cycle instruction set. In *Proceedings of the IEEE First International Tools for AI Workshop*, September 1989.

[Moore, 1985] Robert C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25(1):75–94, 1985.

[Pearl, 1988] Judea Pearl. Embracing causality in default reasoning. *Artificial Intelligence*, 35(2):259–271, June 1988.

[Pednault, 1989] Edwin P. D. Pednault. ADL: Exploringthe middle ground between STRIPS and the situation calculus. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 324–332. Morgan Kaufmann Publishers, Inc., 1989.

[Rao and Foo, 1989] Anand S. Rao and Norman Y. Foo. Minimal change and maximal coherence: A basis for belief revision and reasoning about actions. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 966–971, August 1989.

[Reiter, 1991] Raymond Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, 1991.

[Schubert, 1989] Lenhart K. Schubert. Monotonic solution of the frame problem in the situation calculus: An efficient method for worlds with fully specified actions. In *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer Academic Publishers, 1989.

[Shoham, 1986] Yoav Shoham. What is the frame problem? In *Proceedings of the Workshop on Reasoning about Actions and Plans*, pages 83–98. Morgan Kaufmann Publishers, Inc., 1986.

Case–based Meta Learning: Using a Dynamically Biased Version Space in Sustained Learning

Jacky Baltes and **Bruce MacDonald**
The University of Calgary
2500 University Drive NW
Calgary, Alberta T2N 1N4, Canada
{baltes,bruce}@cpsc.ucalgary.ca

Abstract

It is well–recognized that in practical inductive learning systems the search for a concept must be heavily biased. In addition the bias must be dynamic, adapting to the current learning problem. Another important requirement is sustained learning, allowing transfer from known tasks to new ones. Previous work on dynamic bias has not explicitly addressed learning transfer, while previous case–based learning research suffers from a variety of problems. This paper presents a method of Case–Based Meta Learning (CBML), in which the cases are concepts, rather than instances, and retrieved similar concepts are used as a skeletal version space to speed up learning. CBML is independent of the concept representation language. The CBML–Clerk system, which learns repetitive operating system tasks, is presented as a demonstration.

1 Introduction

Autonomous agents must be able to learn the classifications of objects by induction. For example, an autonomous robot must learn concepts such as "sidewalk" and "cliff," so it can behave safely. An intelligent agent in the operating system domain must be able to learn concepts such as a user–specific class of backup files. This could include file names such as *file*.`BAK`, *file*.`CKP`, *file*~, *file*.~*n*~, and #*file*#, as well as more specific formats (e.g. *file*.`save_21_June_1991` and archived files). Some researchers even argue that learning by induction is the main attribute of an intelligent process [Goldfarb, 1991].

In general, the learning problem is to search the space of all possible concept descriptions for a concept consistent with known examples. As well as the traditional resources measures of time and space complexity, designers of induction algorithms must deal with "sample complexity." The number of examples required to learn a concept, which is related to the user effort in teaching, must be minimized so that the user is not over–burdened [MacDonald, 1991].

In practical systems it is important to make learning manageable by dynamically biasing the concept search in response to the particular examples available, and the learner's current environment [Heise, 1989; Heise and MacDonald, 1991].

This paper argues that case–based meta learning (CBML) provides a powerful and intuitive framework for the implementation of a dynamic bias, and presents the CBML–Clerk, which learns repetitive operating systems tasks from examples. As well as using examples to bias the formation of a concept description, previously learned *concepts* strongly bias the search. It is this latter bias that effectively reduces the sample complexity in terms of the questions a user must answer during learning. As the system learns knowledge about concepts and their descriptions (meta knowledge), it avoids some of the problems associated with case–based learning systems.

After a review of learning in version spaces, inductive bias, and the importance of dynamic bias, the paper introduces CBML, and presents a case study with experimental results that show a decrease in the number of questions required.

1.1 Learning in Version Spaces

A *version space* is the set of all concepts consistent with the given examples and counterexamples [Genesereth and Nilsson, 1987]. If there is a partial ordering on the set of concepts by generality (forming a *version graph*) then the space can be represented efficiently by its upper or general (G) and lower or specific (S) boundary. The Candidate Elimination Algorithm [Mitchell, 1977] is a bi–directional search through a version graph. A positive example forces the S set to be generalized to include it, and removes members of the G set that do not cover it. A negative example forces the G set to be specialized to exclude it, and removes members of the S set that do cover it. The two sets may converge to one "correct" concept. If the boundaries overlap, the version space collapses. This is the result of incorrect example classification or a representation language that is not powerful enough to describe the appropriate concept.

2 Inductive Bias

Both empirical observation and the complexity of inductive learning show that the success of a learning algorithm depends on the method used to restrict the hy-

pothesis space (i.e. the inductive bias) [Mitchell, 1980; Michalski, 1983; Haussler, 1986; Heise and MacDonald, 1991]. Version spaces are an appropriate framework in which to examine bias, particularly dynamic biases, since the candidate elimination algorithm maintains *all* consistent hypotheses, and the result is independent of the order in which examples are presented. Once the version graph is defined, there is no further bias imposed (although the two boundaries move closer as more examples are seen).

Informally, an inductive bias can be defined as any method that prefers one hypothesis over others. Three different categories can be distinguished, based on the implementation mechanism.

- An *absolute* bias is a restriction in the representation language. Certain concepts are not expressible, as they are not in the hypothesis space. A popular absolute bias is the restriction to conjunctive concepts; conjunctions of feature–value pairs.
- A *relative* bias selects one hypothesis over another if both of them are consistent with the example set.
- A *random* bias would select a consistent hypothesis at random.

This paper concentrates on relative biases since they do not prevent the system from learning any concept. Commonly used relative biases are based on the complexity of the concept description or the number of relevant features. For example, given 1, 3, and 8 as positive examples, a learner may prefer the concept "any digit" over "any odd digit or 8." Later if 4 is given as a negative example, the learner must reconsider. However, it is important to note that a fixed relative bias improves the learning only for the initially preferred hypotheses.

Two other dimensions for the comparison of biases are strength and correctness. A *weak* absolute bias does not rule out many hypotheses, whereas a *strong* bias significantly reduces the hypothesis space. A weak relative bias has a minimal effect on the search order, while a strong one reduces the effective search space markedly. A *correct* bias does not rule out the correct concept to be learned, while an *incorrect* one does. Strength is independent of any particular concept, whereas correctness must be defined relative to a concept. A learning system should have a strong bias that meets a certain measure of correctness for the concepts it must learn.

However, a static bias is not generally useful. Heise [Heise, 1989; Heise and MacDonald, 1991] has proposed that the dynamic nature of bias is the primary consideration in the design of real world inductive learners, and shown its usefulness in the ETAR robot system. For example, if a system is trying to learn concepts of file names, it is reasonable to prefer prefixes `test*` and suffixes `*.c` over concepts based on occurrences of single characters such as `*e*`. Many users organize their files by a prefix related to the content of the file (`project1`, `test`, ...) and a suffix indicating its type (`.c`, `.tex`, `.txt`, ...). However, if the same system is to learn concepts in other operating system domains, such as string manipulation in a text editor, it may need to learn concepts such as "punctuation symbol," or "any word with the letter `Z` in it" (`*Z*`). Here the prefix/suffix preferences are inappropriate, although the learner should not discard them; it may need to read in files using the file name concepts it knows. Furthermore, it is conceivable that some users organize their files in a different way, for example placing different file types in different directories, so that preferred concepts might include `project/*/test` or `project/*/a.out`. Therefore, the bias must be dynamic so that the learner can adapt to both the user preferences and the current context.

3 Systems that use Dynamic Bias

STABB Utgoff argues that there are three steps in bias adaption [Utgoff, 1986].

- *Recommend* new concept descriptions that should be added to the hypothesis space.
- *Translate* the recommendations into expressions of the concept description language.
- *Assimilate* newly formed concepts into the original hypothesis space.

STABB represents the concept space as a version graph. An overly strong bias may remove the required concept, and this is detected when the version space collapses (i.e. there are no more hypotheses that are consistent with all examples). STABB then adds extra nodes to the version graph. One heuristic for new node selection is to add the least disjunction required, as depicted in figure 1 where the concept `sin` $\vee$ `cos` is added.

Assimilating the new concept is not straightforward for STABB because adding nodes to a version graph may change the boundaries.

Predictor Gordon's system uses a feature value representation for instances [Gordon, 1990]. An example object in the domain can be represented as

```
object = (mat=wood,size=large,shape=sphere).
```

The Predictor system uses three assumptions to adapt the bias.

- *Irrelevance* is used to *mask* features from the concept description (e.g. ignore the material of an object).
- *Cohesion* determines when to climb a generalization hierarchy of a feature. For example, in the version space of figure 1, if *sin* is the only example, *cohesion* will try to generate the concept description *trig*.
- *Independence* is used to mask feature–value pairs from the representation language. Two features are independent if and only if you can independently change one of the feature values and the resulting object is also a member of the concept. If

  `obj1=(red, block, wood)` and

  `obj2=(green, sphere, wood)`

  are positive examples, then `color` and `shape` are independent if and only if

  `obj3=(green, block, wood)` and

  `obj4=(red, sphere, wood)`

  are positive examples as well.

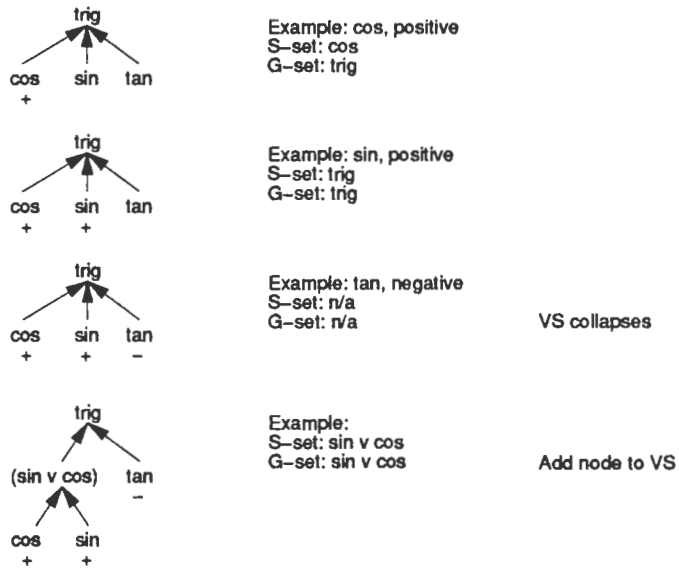


Figure 1: Dynamic Bias in the STABB System. A least disjunction is added when the space collapses.

Predictor tests whether any of the biasing assumptions can be applied and then actively tries to verify this assumption. Although it is an incremental learning system, it can take advantage of the set of examples supplied. For example, if some objects satisfy the *irrelevance* criteria for a given feature, the Predictor system searches the set of examples for instances that verify or invalidate this assumption. If the assumption is verified for the current example set, the bias is adjusted and the hypothesis space reduced by masking the *irrelevant* feature. If in the future other examples show that the feature is not irrelevant, it will be unmasked.

ETAR Heise's system uses dynamic bias to learn robot procedures from examples [Heise, 1989; Heise and MacDonald, 1989]. In contrast to STABB, ETAR strengthens the bias by focusing on aspects of a task trace that meet a relevance criterion. The criterion is spatial locality relative to the robot hand. This enables a raw numeric sequence of teacher guided robot positions to be partitioned into a chain of symbolic action nodes. The bias also enables action nodes to be merged when there are similar nodes within and between example task sequences, thus enabling the determination of branches and loops in the task. Throughout, the bias is driven by the example, or task trace, and dominates the search for an appropriate procedure. Example tasks include blocking stacking, and sorting objects from a conveyor.

4 Issues in Dynamic Biasing

The common approach in the STABB and the Predictor systems is to use heuristics to adapt the representation language and thus the hypothesis space. ETAR on the other hand uses its bias to modify the input example information, and the search process. All three address the problem of learning a single concept, not taking advantage of previous learning in a new learning episode.[1] This has two limitations. In an autonomous agent, rather than defining different representation languages and different biases for each concept, we would like to implement a general purpose learning algorithm that supports transfer of concepts. Second, the choice of initial bias has been ignored. The STABB system starts out with a strong bias and weakens this bias if forced to. Predictor uses a weak bias and tries to strengthen it. This means that both systems require extra work to achieve a successful bias. A system should use previous experience to approximate the initial bias.

The systems' biases operate directly on particular representation languages. If the language is changed, for example from DNF to CNF, then the dynamic biasing algorithm must be updated. In addition, since transformations of the representation language are made explicitly in STABB and Predictor, these approaches require explicit adaption rules. Such rules may be hard to find and to compute for non–trivial representation languages. For example, the Predictor's *independence* biasing assumption is hard to represent in DNF; the concept for independent `color` and `shape` in the description above would be:

`(red and block) or (red and sphere) or (green and block) or (green and sphere).`

The representation is simpler in CNF: `(red or green) and (block or sphere)`.

CBML enables task transfer, and is not specific to particular representation languages.

5 Case–based Meta Learning

In case–based learning (CBL) classified examples (cases) are stored and used to help in classifying new, unknown examples [Aha, 1991]. The focus is on different issues than other case–based reasoning methods. The case

[1] Although Heise's current work addresses this (personal communication).

representation is restricted to feature–value pairs and cases are not adapted to fit a new situation. A simple indexing scheme uses a similarity assessment of the new and previous cases. Approaches range from encoding a large amount of domain knowledge (e.g. Protos [Bareiss, 1989]) to computing similarities dynamically (e.g. MBRtalk [Stanfill and Waltz, 1988]).

Breiman *et al.* [1984] argue that simple CBL algorithms are computationally expensive (because similarities between all cases and the current concept must be computed), intolerant to noise and irrelevant features, sensitive to the similarity function, give no simple way to define similarity functions for symbolic–valued features, and provide little information about the structure of the data. Aha [1991] proposes methods to overcome the first two, and notes that CBL must be sensitive to the current context. GCM–ISW [Aha and Goldstone, 1990] uses context as well as goal features in the similarity assessment of new cases.

A major criticism of CBL is the large amount of storage required. Recent research has alleviated this problem by storing only instances that can discriminate among different classifications [Aha, 1991]. In this paper cases do not represent single instances, but concepts learned in previous tasks. The case memory grows only linearly with the number of tasks that the system learns. The justification of this approach is analogous to Hammond's justification for case–based planning [Hammond, 1989]. Learning is an expensive operation so the results of the learning procedure should be stored and reused in the future.

In CBML previous cases are used as a skeleton of the hypothesis space, to guide the search. The skeletal hypothesis space consists of concepts that have been successfully used in the past on a similar task. Once the most appropriate known concept is found, ordinary learning algorithms are used to find the correct node.

CBML implements a bias relating a new concept with previous learning experiences and tries to maintain the partitioning of the current instance set that would be imposed by previously know concepts. The bias is to maintain sets of instances that were learned by previous similar cases. For example, if each previous case generates one consistent classification (positive or negative) for all elements of the current instance set matching `*.txt`, then the preferred hypotheses are those that assign one particular classification to all strings ending in `.txt`.

One distinction between previous dynamic biasing systems and CBML systems is that CBML bias is adjusted only when learning a concept in a similar task. It cannot yield better performance if the system is used only to learn one specific task. However, CBML and other dynamic biasing algorithms can complement each other since CBML is independent of the specific learner/classifier algorithm. Furthermore,

1. CBML supports context dependent biases. Previous concepts that were useful in executing similar tasks are retrieved. Different tasks can use a common representation language.
2. CBML can provide an initial bias. If all similar previous concepts assign the same classification to all strings matching `*.txt`, then the important feature of this task may be suffixes.
3. CBML does not change the representation language explicitly, is independent of the language and the concept learner/classifier, and needs no knowledge of how to change the bias to focus on, say, suffixes.
4. Since transformations are made only implicitly, transformation rules are not necessary.

CBML also does not suffer from the problems usually associated with CBL. The computational complexity is reduced since only concepts and not instances of concepts are stored. Noise in a CBML system is equivalent to retrieving a case that uses a different bias than the concept to be learned. This affects the efficiency but not the correctness of the learning procedure. Instead of retrieving one similar case, CBML systems retrieve all similar cases. Therefore, a CBML system is more robust with respect to irrelevant features and the choice of the similarity function. A CBML system uses a learner/classifier routine for learning and thus does not need to represent symbolic-valued features or structured data at an instance level.

6 Implementation of the CBML–Clerk

To show the operation of CBML in a realistic learning setting, Baltes implemented a CBML system on top of the Shell–Clerk [Baltes, 1991], which is an instructable system (see [MacDonald, 1991]) that learns repetitive operating system tasks by example, such as copying files, arranging mail messages, or reading news articles. File names are represented as strings and the representation language is a subset of regular expressions, limited to at most three terms in a disjunction. Nevertheless, it can learn concepts such as all backup files (`*~ or #*# or *.CKP`), all C source files (`*.c or *.h`), and all test files (`*test* or *Test* or *TEST*`). The Shell–CLERK uses a "symmetric version space" (SVS) approach to learning string concepts, but requires a few too many questions (19 for 96 files) to learn common concepts such as `*.*` or `*~`. Based on the argument given in the previous section, CBML–Clerk was designed as an extension.

Instead of using a dynamic bias to select concept descriptions and use these concepts, constraints imposed by the domain require that the learning algorithm must not over–generalize (mistaken file erasure is unacceptable). The CBML–Clerk uses the instance set partitions generated by previous tasks as a way to reduce the number of questions to the user. It will ask for the classification of one of these sets, rather than that of a single example. Although there is no decrease in the number of instance classifications required from the user, less work is required to provide these classifications.

The Shell–Clerk requires the user to begin learning by listing a set of files that contains the target ones, then inputing a command and a prototype file name on which this command is to be executed. The original version then proceeded to question the user about individual file names. The CBML–Clerk begins the same way, then

proceeds to ask questions about possibly relevant sets of file names.

6.1 Case Representation

A case records the prototype file name, the command, and the concept description produced by the SVS algorithm. The candidate elimination algorithm can not be used directly because the G set is possibly infinite for a representation language with limited disjunctions in an infinite domain.[2] Concepts are described by a positive *cover set* — the most specific description of all positive instances — and a negative *cover set* — the most specific description of all negative instances — plus other information. For example, a concept matching strings `a`, `b`, and `c` but not any digits or other lower case characters, is represented by the following structure[3]

```
Pos-Cover: (a or b or c)
Neg-Cover: (<digit> or <lowercase>)
```

6.2 Case Indexing

The CBML system retrieves all similar cases when trying to learn a new concept. These concepts define the abstract hypothesis space. A case is considered similar to the current situation if at least one of the following conditions holds:

1. The concept uses the same prototype string. The SVS algorithm uses the first example string as a prototype to determine the number of independent disjunctions in a concept.
2. The user executed the same command on the strings matching the concepts. For example, let us assume that the user taught the system a task where all strings matching the concept had to be copied to a different directory. In a new task, the user tells the CLERK to copy all files matching a new concept, so the original concept will be retrieved.

6.3 Case Adaption

CBML provides the opportunity to adapt old cases to new situations, because cases are concept descriptions that may be used to partition the instance set. For example, if the concept to be learned (say `*.txt`, although the learner does not know this yet) and a retrieved concept (say `*.tex`) are determined to be inconsistent, an adaption rule can be fired to change the recalled concept description. In CBML–Clerk, if a similar case is inconsistent then the learner examines the complement of the concept, and if this is also inconsistent, then that retrieved concept is removed from consideration.

6.4 Case Storage

One major criticism of case–based learning systems is the large amount of storage required to store all instances. Recent research has alleviated this problem by only storing instances that can discriminate among different classifications. Only instances that were wrongly classified are added to the case memory [Aha, 1991].

[2]See [Baltes, 1991] for details of the SVS algorithm.

[3]The SVS algorithm requires two extra *cover sets*. These can be safely ignored for the discussion.

Since CBML stores only concepts and not instances in its case memory, this problem does not arise. It stores only one new concept for each new task that it learned. Therefore, the size of the case memory grows only linearly with the number of tasks that the system can perform.

6.5 Learning using CBML

The input to CBML–Clerk is the current command–prototype pair, a set of instances that must be classified to perform the given task, a memory of previous concepts, and a learner/classifier algorithm. The learner/classifier routine classifies instances as *positive*, *negative*, or *unknown*. If provided with the correct classification (*positive* or *negative*) of an *unknown* instance, it returns an updated concept that is able to classify the previously *unknown* instance in the future.

Table 1 describes the CBML algorithm for the Clerk. After retrieving all similar cases from the concept memory (Step 1), the classification of all instances as well as all similar cases is computed (**for-each** beginning Step 2). If none of the instances in the instance set are *unknown*, then there is no need to learn. The algorithm simply returns after adding the concept to the case memory (Step 4). Similar cases are removed if inconsistent with the current instance set (Step 3).

In the next stage, similar cases are updated with the information from the current instance. For example, if case c is unable to classify instance i, but the current concept classifies i as either *positive* or *negative*, case c is updated to include the classification of i. Note that this update is temporary and is not reflected in the case memory (Step 5).

The algorithm next removes decision equivalent concepts (Step 6). In the example of table 2 the concepts `*.c` and `*.c or *.cc` will be combined because they are decision equivalent for the given set of new instances.

In the next step, the system is trying to find a subset of the instance set that allows it to discriminate best among all similar cases. For this purpose, a two–dimensional array is constructed, called the Min–Max–Table (Step 7). The rows of the Min–Max–Table consist of sets of instances that have the same classification under all similar cases and are now classified as *unknown*. If there is only one element in the instance set for a given row, the row is deleted from the Min–Max–Table because it will only require one question to the user (at a later stage) to find the classification of this instance. The columns consist of the similar concepts. The entry at row i and column j is the classification of instances in set i under the concept j. An example of a Min–Max–Table is given in table 2.

All instances belonging to the set with the best Min–Max–Value are then presented to the user with the request to classify *all* of the instances as *positive*, *negative*, or *neither*. If the answer is *positive* or *negative*, the concept to be learned is updated with the new classification for all elements of the set. If the user answered *neither*, these instances are removed from the instance set (Step 8). The user will be asked about the correct classification for all these instances later (Step 11).

```
CBML(command,prototype-string,instance-set,memory,learner/classifier)
        (returns an updated case memory)

1.    similar-cases := find-similar((command,prototype-string), memory)
      new-concept := nil-concept
      unknown-instances := nil

      while (similar-cases not empty)
2.       for-each case in similar-cases
3.          if inconsistent(instance-set, case) then
               remove(case, similar-cases)
            case' := make-case(command,prototype-string,concept(case))
            (_, class) := learner/classifier(case', instance-set)
4.          if all-instances-classified(class) then return(update-memory(case'))
5.          replace(similar-cases,case,update(concept(case),class))
6.       remove-equivalent(similar-cases)
7.       bmm := best-min-max(similar-cases)
8.       if ask-user(instances(bmm)) = TRUE or FALSE
9.          then new-concept := update(new-concept,instances(bmm))
10.         else unknown-instances :=
                    append(unknown-instances,
                           remove(instances(best-min-max), instance-set))

11.   (new-concept,_) := learner/classifier(
                           new-concept,
                           append(unclassifiable-instances(instance-set),
                                  unknown-instances))
      return(update-memory(new-concept))
```

`learner/classifier` returns a new concept and the c' sification partition made by it. `concept(case)` returns the string concept of the case. A case includes a command, prototype string and a string concept.

Table 1: The CBML Algorithm

The heuristic used to select the best Min–Max–Value is based on the assumption that the response of the user will often be *positive* or *negative*, so that a large number of similar cases will be removed. Let us assume that a set of instances is classified as *positive* by two similar cases, as *negative* by three cases, and as *unknown* by one case. Let us further assume that the size of the set of instances associated with this row in the Min–Max–Table is three. If the user's classification is *positive*, three cases can be removed from the Min–Max-Table, because they are inconsistent with this information. On the other hand, if the classification is *negative* only two cases can be removed. If the user answers *neither*, no concept can be removed and the user must be asked for the classification of these instances separately.

Min–Values and Max-Values are calculated for each row. The CBML algorithm selects the next instance set according to these rules:

- Choose the maximum of the Min–Values of all rows. The Min–Value of a row is the minimum of the number of *positive* classifications and *negative* classifications in a row.
- If there is a Min–Value tie among rows, select the instance set (row) with the highest Max–Value. The Max–Value is the maximum of *positive* and *negative* classifications.

For example, in table 2, instance set `test.o`, `test.dat`, `a.out` will be selected. The set `test.h` has a Min–Max–Value of (1,1)[4] while each other set has a Min–Max–Value of (1,2). So the largest set is selected.

If there is more than one similar case left, the algorithm loops back to Step 2.

7 Results

The CBML–Clerk was tested on a sequence of different tasks in the operating system domain. Common concepts such as `*.*`, `*.c`, `*.tex`, `*~` were learned after only a few example tasks. The example concept in table 2 is learned after only three questions. Table 3 shows the results for the first five concepts in the original SVS paper [Baltes, 1991]. Here the concepts are not learned in any particular order and are not that well related to a particular task, so that CBML is not particularly effective. Table 4 shows a more significant improvement when concepts are related, and are taught in a reasonable order. In both tests the command was the same for all concepts.

[4] As above, this row will be deleted since the instance set includes only one concept.

Learned concepts	`*.* *.c (*~ or *.bak) (*.c or *.cc)`
New Concept	`*.c or *.h`
New Instances	`test.c test.c~ test.bak test1.c`
	`test.h test.o test.dat a.out`

	new	*.*	*.c	*~ or *.bak
test.c	?	t	t	f
test1.c	?	t	t	f
test.c~	?	t	f	t
test.bak	?	t	f	t
test.h	?	t	f	?
test.o	?	t	f	f
test.dat	?	t	f	f
a.out	?	t	f	f

Table 2: Example of the CBML–Clerk Min–Max–Table.

Concept	Prototype	CBML Q	SVS Q	Comments
test* or #test*	test.ss	32	32	identical to SVS
*	generic.c	2	19	optimal case for CBML
*(file-io or built-in)	built-in_amiga.ss	38	38	does not match previous
(_amiga or _sun)*				prototype; not adapted
(amiga or sun)	built-in_amiga.ss	34	42	
e	test.ss	39	41	

Table 3: Results for the first five concepts reported in [Baltes, 1991]. There are 96 files and the third and fourth columns give the number of questions asked by CBML and the original SVS algorithm.

Concept	Prototype	CBML Q	SVS Q	Comments
*.c	test.c	8	8	For perfect learning it must ask about all
.	test.c	2	8	an optimal case
a.out	a.out	2	8	
*~ or *.bak	test.bak	6	8	
*.c or *.h	test.c	6	8	

There were eight files:
`a.out test.c test.h test.c test.bak test1.c test.o test.dat.`

Table 4: Results for some related concepts, again comparing CBML and the original SVS algorithm.

8 Conclusion

This paper argues that skeletal, case–based hypothesis spaces can be used to reduce the sample complexity of learning algorithms. CBML is an simple, efficient, and intuitive way to construct hypothesis spaces from previous experience. By using a case–based approach, the constructed spaces are context dependent and can therefore be used as a dynamic bias.

CBML overcomes difficulties associated with dynamic biasing and case–based learning. An advantage is its robustness, which is gained by retrieving *all* similar cases. Furthermore, CBML does not require an explicit set of transformation rules for adapting bias. CBML is independent of the concept representation language. Although we have yet to devise sophisticated case adaption rules, CBML enables learning without case adaption. Improvements could also be made to the simple indexing scheme. Furthermore, at the moment only a single skeletal version space is generated. It seems reasonable that a hierarchy of spaces could further reduce the search through the hypothesis space, just as abstraction hierarchies do in planning. The algorithm could also be extended to combine similar cases in the case memory, if many are found. This would reduce the computational complexity as well as the storage requirements.

CBML promises to be an appropriate trade-off between the detail of case–based learning, and the generality of inductive inference.

References

[Aha and Goldstone, 1990] D. W. Aha and R. L. Goldstone. Learning attribute relevance in context in instance–based learning algorithms. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 141–148, Cambridge MA, 1990. Lawrence Birnbaum.

[Aha, 1991] David W. Aha. Case–based learning algorithms. In *Proceedings: Case–Based Reasoning Workshop*, pages 147–158, 1991.

[Baltes, 1991] Jacky Baltes. A symmetric version space algorithm for learning disjunctive string concepts. In *Proc. Fourth UNB Artificial Intelligence Symposium*, pages 55–65, Fredericton, New Brunswick, September 20–1 1991.

[Bareiss, 1989] Ray Bareiss. *Exemplar–Based Knowledge Acquisition*. Academic Press, Inc., 1989.

[Breiman *et al.*, 1984] L. Breiman, J.H Friedman, R. A. Olshen, and C. J. Stone. Classification and regression trees. Technical report, Wadsworth International Group, Belmont, CA, 1984.

[Genesereth and Nilsson, 1987] M. R. Genesereth and N. J. Nilsson. *Logical foundations of artificial intelligence*. Morgan Kauffman, 1987.

[Goldfarb, 1991] Lev Goldfarb. Verifiable characterization of an intelligent process. In *Fourth UNB Artificial Intelligence Symposium*, pages 67–80, 1991.

[Gordon, 1990] Diana Faye Gordon. *Active Bias Adjustement for Incremental, Supervised Concept Learning*. PhD thesis, University of Maryland, 1990.

[Hammond, 1989] Kristian J. Hammond. *Case Based Planning*. Academic Press Inc., 1989.

[Haussler, 1986] David Haussler. Quantifying inductive bias in concept learning. Research report UCSC-CRL-86-25, University of California, Santa Cruz, CA 95064, November 1986.

[Heise and MacDonald, 1989] Rosanna Heise and Bruce A. MacDonald. Robot program construction from examples. In *Proc. National Irish AI Conf.*, Dublin, Ireland, September 1989. Also in book form, edited by A. F. Smeaton and G. McDermott (Eds.), AI and Cognitive Sciences '89, Springer–Verlag, 1990.

[Heise and MacDonald, 1991] Rosanna Heise and Bruce A. MacDonald. Dynamic bias is necessary in real world learners. Technical report, University of Calgary, 1991.

[Heise, 1989] Rosanna Heise. Demonstration instead of programming: focussing attention in robot task acquisition. Master's thesis, Department of Computer Science, University of Calgary, 1989.

[MacDonald, 1991] Bruce A. MacDonald. Instructable systems. *Knowledge Acquisition*, 3(4):381–420, December 1991.

[Michalski, 1983] R. S. Michalski. A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonnel, and T. M. Mitchell, editors, *Machine Learning, Vol. 1*. Tioga, Palo Alto CA., 1983.

[Mitchell, 1977] Tom M. Mitchell. Version space: A candidate elimination algorithm approach to rule learning. In *Proceedings of the 5th International Conference of on Artificial Intelligence*, pages 305–310, 1977.

[Mitchell, 1980] Tom Mitchell. The need for biases in learning generalizations. Technical Report TR CBM-TR-117, Rutgers University, 1980.

[Stanfill and Waltz, 1988] C. Stanfill and D. Waltz. Learning to read; a memory–based model. In *Proceedings of a Case–Based Reasoning Workshop*, pages 402–413, 1988.

[Utgoff, 1986] Paul E. Utgoff. *Machine Learning of Inductive Bias*. Kluwer Academic Publishers, 1986.

Learning Expertise from the Opposition*

Susan L. Epstein
Department of Computer Science
Hunter College and The Graduate School of The City University of New York
695 Park Avenue, New York, NY 10021
USA

Abstract

For very difficult games, like Go, it is increasingly clear that the most competitive programs will be those whose expertise is developed through learning during competition. This paper explores how the nature of the opposition during training affects the quality of learned behavior in two-person, perfect information board games. It considers different kinds of competitive training, the impact of trainer error, appropriate metrics for post-training performance measurement, and the ways those metrics can be applied. Variations in the playing skill learned from many kinds of opposition are described here for three different games. The results argue for a broad variety of training experience with play at many levels. This variety can either be driven by inherent elements of chance in the game or be introduced deliberately into the training. A case is made for extensive, thoughtful training of systems that learn, and for cautious reliance upon them.

1. Introduction

Educators promulgate many philosophies about what makes a good learning situation for humans, but it is difficult to compare how the same individual learns the same skill in more than one environment; prior learning experiences are to some extent ineradicable. With a computer program, however, it is possible to learn from the beginning as often as one likes, to compare and contrast learning environments in a variety of situations, and to test the resultant skill extensively without permitting further learning. In particular, with machines instead of people, one can ask how the nature of the opposition determines what, and how quickly, a game player learns.

The thesis of this paper is that the acquisition of absolute expertise in a competitive domain demands a broad variety of challenging experience as well as more thorough testing than traditionally anticipated. The contribution of this paper is its analysis of the impact of the training environment on learning to play games. It includes the formulation of appropriate performance metrics and recommendations for training a program to play games.

*This work was supported in part by NSF 9001936 and PSC-CUNY 668287.

2. Competitive Learning and Expertise

Traditional AI game playing programs, like Deep Thought and HiTech, use fast, deep search to identify relevant future positions and evaluate their strength [Anantharaman *et al.*, 1990; Berliner & Ebeling, 1989]. These programs rely on special-purpose hardware, clever storage and retrieval tactics, a few well-known search heuristics, and raw computing power to search deeply and quickly. There is a growing consensus in the AI game-playing community, however, that a game like Go cannot be played as well as chess is with such techniques, because Go's search space is so much larger than that of chess, and because Go offers so many more possibilities at each choice point.

As a result, there has been substantial recent interest in programs that learn to play games. Samuel's Checker Player was an early effort that learned an evaluation function based on input features of the checkers board [Samuel, 1963, 1967]. TD-gammon learns to play backgammon with a neural net that, after much practice, holds its own against a world master [Tesauro, 1992]. Morph learns to play chess with a pattern cache that is gradually improving against a strong commercial chess program [Levinson and Snyder, 1991]. Hoyle learns to play many simpler two-person perfect information board games extremely well against a variety of experts [Epstein, 1992]. TD-gammon learns the weights for its neural net, Morph learns patterns, and Hoyle learns useful knowledge about each game, knowledge that is probably correct and possibly applicable in a variety of contexts.

Do game-learning programs learn to play perfectly, or only as well as people? Against what kind of opposition do they learn to play best? Does the nature of the opposition affect their learning speed or long-term memory requirements? How does learning differ when the opposition's errors are due to lack of foresight, to lack of knowledge, or to random decisions? This paper describes a recent experiment with Hoyle to address these issues. Because Hoyle learns a broad variety of games against any specified opposition, it can be used to explore whether the answers to these questions are game-dependent.

3. Experimental Design

Each *trial* for this experiment has Hoyle learn a game while playing against another program, called a *trainer*, and then tests Hoyle's post-learning playing skill against four kinds of opposition.

3.1 The learning program

Hoyle is based upon FORR, a general architecture for a learning and problem solving expert, one that postulates and capitalizes upon regularities (Epstein, 1991). Hoyle's domain is two-person, perfect information board games. Given the definition of a new game, Hoyle begins as a rule-abiding novice that plays against an external, presumably expert, model. This model is only observed, never queried. As Hoyle plays it gradually improves, often becoming expert or even perfect at a game.

Each *game* Hoyle can play is an instantiation, a pre-specified, input instance, of a *game frame*. The only specific knowledge Hoyle has about a new game before playing is the values associated with these slots. Some slots hold constants: the name of the game, the markers assigned to each participant, the initial state of the board before play, whether the board is two-dimensional or three-dimensional, how often to scroll the screen during play, which places on the board are considered adjacent in games where pieces may slide, which lines on the board are considered wins if the game is won that way. Other slots hold the names of LISP functions: they display the current game on the screen, read and filter input moves, generate and effect legal moves, detect the end of a contest and who has won, and transform the board back and forth between a list and a visual representation. These functions are very brief, typically a total of less than 100 lines of code per game.

Hoyle's *game-playing algorithm* is a script that provides pre-defined, uniform, procedural direction to the program. The game-playing algorithm enables Hoyle to perform as if it were experienced in game playing, without expertise at any particular game. This script detects when it is the program's turn to move, ensures that the participants alternately make legal moves, and announces the end of each contest, along with any winner. Given a valid game definition and the game-playing algorithm, Hoyle simulates a rule-abiding novice, one that makes legal, if not astute, moves.

The game-playing algorithm also triggers Hoyle's Learner. The *Learner* is a set of algorithms for the discovery of *useful knowledge*, knowledge that is expected to be relevant and may be correct. Based on its playing experience, Hoyle computes and stores game-dependent useful knowledge. The Learner has a uniform, heuristic, game-independent learning procedure for each item of useful knowledge. If the Learner were to retain everything Hoyle experiences, useful knowledge for an interesting game could quickly become unmanageably large. Therefore the learning algorithms generalize and are highly selective about what they retain. There are useful knowledge slots to record average contest length, applicable two-dimensional symmetries, good openings, moves that expert opposition appears to have found valuable, relevant forks, important contest histories, whether going first or second is an advantage, and *significant states*, situations that will inevitably be won or lost when both participants play expertly.

The application of learned useful knowledge is the task of Hoyle's Advisors. An *Advisor* is a heuristic that makes comments about legal moves when it is the program's turn to make one. A *comment* is the Advisor's name, a move, and a *weight*, an integer from 0 to 10, indicating an opinion somewhere in the spectrum from strong aversion (0) to enthusiastic support (10). Each Advisor constructs its comments based upon the current state and the useful knowledge for the current game. For example, Victory compares useful knowledge with the current legal moves, and recommends with a weight of 10 each legal move that results in an immediate win.

Whenever it is Hoyle's turn to move, the game-playing algorithm provides the Advisors with the current game state, the legal moves, and any useful knowledge about the game already acquired. (If Hoyle has had little or no experience at this particular game, there may be no useful knowledge.) From the Advisors' comments, a simple arithmetic calculation selects a move that is forwarded to the game-playing algorithm for execution.

3.2 The trainers

There are five kinds of trainers in this experiment: self, random, perfect, fallible, and informed. The *self trainer* is the learning program itself, which takes both sides in every contest. The *random trainer* makes randomly-chosen legal moves, has no knowledge, and treats every possibility as equally likely. The *perfect trainer* plays as if it had exhaustively searched the tree and minimaxed the result back up to the current state to select its next move [Nilsson, 1980]. If there is more than one equally good move, the perfect trainer will make a random choice from among them. A perfect trainer for tic-tac-toe, for example, opens half the time in the center and half the time in a randomly-chosen corner. A perfect trainer is designed to provide a variety of high-quality expert play without mistakes. The *fallible trainer* is a variation on the perfect trainer, a variation that periodically has the opportunity to make a mistake. The fallible trainer with an error rate of $e\%$ makes the perfect trainer's move $100 - e\%$ of the time, and makes a randomly-chosen, legal, not necessarily imperfect move $e\%$ of the time. There is a spectrum of fallible trainers in this experiment with e values in multiples of 10, from 10 to 90. Although a 10% chance of error may seem high, in many games the *branch factor* (number of legal alternative moves) decreases as play progresses, so that the likelihood of an error diminishes towards the end of a contest. For example, if there are three legal moves left in a contest, only one of which is correct, a random selection still has a 33% chance of making the right one. Thus there is only a $.1(.67) = .067$ chance of making a mistake at that point even though $e = 10$. The *informed trainer* applies an input, game-dependent evaluation function in an alpha-beta search to a fixed depth [Nilsson, 1980]. With a depth of d, the informed trainer uses the evaluation function to examine all the relevant nodes d moves after the current state, and minimaxes the result back up to the current state to select its next move. When an informed trainer makes a mistake, it

is because the evaluation function is an approximation of the knowledge inherent in the game tree, not because the trainer has made a randomly chosen move. Informed training is flawed by lack of foresight and lack of knowledge; fallible training is flawed by random error. Each evaluation function is absolutely correct at the end of a contest and reasonable but imperfect elsewhere, as if the trainer had good but incomplete understanding of the game. The informed trainers in this experiment, each with its own trial, have d values from 2 through 8.

3.3 The challengers

Post-learning playing skill is thoroughly tested against varied opposition called *challengers:* a perfect challenger, an expert challenger, a novice challenger, and a random challenger. The four offer a broad variety of competitive experience. The *perfect challenger* uses the same algorithm as the perfect trainer. The *expert challenger* simulates an expert equivalent to a fallible trainer with a 10% error rate; 90% of the time it plays flawlessly, 10% of the time it may err. The *novice challenger* simulates an expert equivalent to a fallible trainer with a 70% error rate; only 30% of the time is it guaranteed to play perfectly. Error rates for both the expert and the novice were selected from laboratory observation of their resultant play quality. Finally, the *random challenger* makes random legal moves.

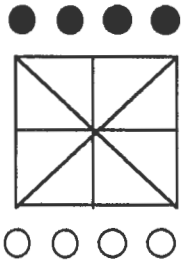


Figure 1. The initial state for achi.

3.4 The games

The three games in this experiment are tic-tac-toe, lose tic-tac-toe, and an African game called achi. *Tic-tac-toe* is played on a three-by-three grid. One participant has five X's, and the other has four O's. Initially the board is empty, and X moves first. A turn is placing one of your markers in any empty square. The first one to place three of the same markers in a row, vertically, horizontally, or diagonally, wins. There are eight such winning lines; play ends in a draw when there are no more empty squares. *Lose tic-tac-toe* is played the same way as tic-tac-toe, except that the first one to place three of the same markers in a row, vertically, horizontally, or diagonally, loses. *Achi* is played on the board in Figure 1. The first participant has four black markers; the second has four white ones. Initially the board is empty, black moves first, and a turn is placing one of your markers on the intersection of two or more lines; there are nine such positions. Once all four of your markers are on the board, a turn is moving one of your markers to the single empty position. The first one to place three of the same markers in a row, vertically, horizontally, or diagonally, wins. There are eight such winning lines. Play ends in a draw when it cycles through the same state for the fourth time.

The construction of a perfect trainer and fallible trainers requires a complete and correct *perfect play theory* for a game, a real-time algorithm that identifies the best possible move from every possible game state. The construction of a thoughtful but partially flawed evaluation function requires good human understanding of the features of a game. The games for which people have both a perfect play theory and such understanding are fairly simple. One way to make the learning task more difficult is to select games where the perfect play theory known to humans is not naturally expressible in the learning program's representation. Lose tic-tac-toe was chosen because Hoyle cannot represent its perfect play theory explicitly, although it can eventually acquire enough useful knowledge to play it perfectly. Lose tic-tac-toe was also selected because a randomly chosen move is likely to be a fatal error, and because the object is to *avoid* achieving a pattern, unlike the other two games. Achi was chosen because of the contrasts it offers to the other two: it has two stages with different move types, it is cyclic, and its branch factor remains four throughout the second stage of every contest. All these games have certain commonalities that make comparison appropriate: their boards are isomorphic and each is known to be a *draw game* (when played by perfect participants a contest always ends in a draw).

3.5 The learning and testing cycle

A *trial* in this experiment consists of a learning experience followed by a testing experience. At the beginning of a trial, Hoyle has no specific knowledge about any of the games. A *learning experience* is determined by the choice of a game and a trainer. Since there are three possible games and, with the values for e and d, 19 trainers, there are 57 trials. Once the game and the trainer for a trial are specified, Hoyle plays a tournament of contests at the specified game with its designated trainer. After the program is judged to have learned to play, learning is turned off, and Hoyle's skill is evaluated in a *testing experience,* a 20-contest tournament against each of the challengers. In both training and testing tournaments, Hoyle and its opposition alternate playing first.

This design makes several assumptions. The learner is not required to discover any role advantage inherent in the tree; the program is told that these are draw games. The program is instructed to stop learning when it meets a *behavioral standard,* i.e., when it draws or wins 10 consecutive contests; it evaluates its playing performance based on this externally specified standard. Finally, the program is expected to perform well against any competition. It should be able to exploit its opposition's errors and to deal well with foolish moves.

Every learning experience is non-deterministic because the trainer or Hoyle can make random legal move choices from time to time. A single run for a trial may therefore not be representative of the trainer's impact on learning performance. To compensate for this uncertainty, each trial is run five times, and the results averaged.

3.6 The evaluation criteria

Measuring whether or not a program plays perfectly requires either exhaustive testing (the play of all possible contests) or a perfect play theory for the game. For most interesting games neither of these is an option. An alternative standard is to require that the program achieve the best possible *outcome* (win, loss, or draw) the game tree offers, from any game state, against any opposition. Evaluation of learning based on contest outcome seeks effective play, rather than perfect play. It still demands, however, that the participant take full advantage of the opposition's mistakes, and identification of those mistakes is, once again, difficult to measure.

A more workable standard is *role performance*. In a draw game, role performance requires a draw whether one moves first or second in the contest, even against a perfect player. Since losses in a draw game are always avoidable, a loss by the learner indicates imperfect performance. A win by the learner in a draw game, however, only indicates the successful exploitation of a fatal error made by the opposition. This experiment uses role performance as the learning criterion: for a draw game and a challenger, *reliability* (the ability to withstand the competition) is measured by the percentage of wins and draws, and *power* (the ability to exploit the opposition's errors) is measured by the percentage of wins. A perfect player would be 100% reliable against all challengers and maximize its power as circumstances permitted. In a draw game, power against the expert challenger is always 0%.

To compare training behaviors across trainer type, construct a similarity metric for post-learning playing behaviors as follows. Let a trial of the experiment be represented as $\mathbf{B} = | b_{ij} |$, a 4×3 matrix where b_{ij} is the average of the jth outcome (number of wins, losses, or draws) against the ith challenger (perfect, expert, novice, or random). Define the *challenger difference measure* of two trials **B** and **B'** for the ith challenger to be the sum of the squares of the corresponding differences in their relevant rows

$$\sum_{j=1}^{3} (b_{ij} - b'_{ij})^2$$

and define the *difference of two testing behaviors* as the sum of their four challenger differences,

$$\sum_{i=1}^{4} \left(\sum_{j=1}^{3} (b_{ij} - b'_{ij})^2 \right).$$

Identical testing behaviors have a zero difference in each row and a zero difference overall. The testing behavior among $\{\mathbf{B_1}, \mathbf{B_2}, \ldots, \mathbf{B_k}\}$ *most similar* to behavior **B** is the one whose difference from **B** is a minimum.

Finally, a learning program with an imperfect trainer may find that learning takes longer, or that it is burdened by many unimportant recollections. For each trial *learning time* is measured by the number of contests to meet the behavioral standard, in this experiment a minimum of 10. *Learning space* is measured by the additional memory allocated to Hoyle's associated useful knowledge cache, a heuristically restricted set of game states, moves, and contest histories recorded from playing experience.

4. Results and Discussion

The results of the 57 trials are summarized here. An important idea is that learning during training is often *incomplete*, that is, that a program can meet the behavioral standard (appear to have learned to draw consistently) without knowing how to play perfectly, or even very well. For example, after drawing 10 consecutive contests of lose tic-tac-toe against a perfect trainer, Hoyle then lost 12% of its contests against the expert challenger. This indicates that learning was incomplete, i.e., that training against a perfect player inadequately prepared the program for competition against a strong player that makes occasional mistakes. In the discussion that follows, Cox and Stuart's non-parametric binomial test for trend is used to calculate whether or not there is a correlation between two sequences of numbers [Conover, 1980]. All correlations cited are 93.75% or better unless otherwise indicated. When one of the sequences is e values, data from the perfect trainer ($e = 0$) and the random trainer ($e = 100$) are also included.

4.1 Individual games

4.1.1 Tic-tac-toe

Hoyle's Advisors are immediately able to support a fairly high level of play at tic-tac-toe. Against any informed trainer, for example, they never lose a contest. There is, however, a game state involving a simple fork from a corner opening, where Hoyle, before learning, will always make an incorrect move and lose the contest. The perfect trainer periodically presents this game state; Hoyle fails, learns from its loss, and never makes a mistake in that state again. During training, however, there is no guarantee when or if that state will arise, particularly with a less than perfect trainer. Hoyle can learn many other things from less high-quality mistakes during training, but this particular piece of useful knowledge is essential for perfect reliability.

Only when $e = 10$ was perfect reliability at tic-tac-toe achieved consistently, i.e., against all the challengers in every run. In every other trial, even in perfect training, Hoyle lost a contest to some challenger in at least one run. These losses ranged from 2% to 5.5% in trials where some other run achieved perfect reliability. This suggests that at best $e = 10$ can be trusted to develop reliability. Reliability against the perfect challenger, the expert challenger, and the novice challenger were negatively correlated with e value, i.e., *the more fallible the tic-tac-toe trainer, the less reliably Hoyle performed.*

Hoyle's power in tic-tac-toe ranged from 12% to 23% against the expert, from 72% to 85% against the novice, and 85% to 98% against the random challenger. Maximal power against the random challenger was developed from training with $e = 60$, against the novice with $d = 6$, and against the expert with random training.

Learning time averaged roughly 12 contests for $e \leq 70$ and for perfect training; in all the other trials it was an overconfident 10. Learning time was negatively correlated with e value; the more fallible the trainer, the faster the behavioral standard was met. Learning space ranged from 1 unit, for self training and all informed training, to 24.6 for $e = 70$. Learning space was positively correlated with e value;

the more fallible the trainer the more, presumably useful, knowledge was acquired. The self-trained program lost 30% of its contests to the perfect challenger and 2% to the novice challenger, but also managed to win 14% against the expert, 72% against the novice, and 93% against the random challenger.

4.1.2 Lose tic-tac-toe

Lose tic-tac-toe is a game where relatively few moves are optimal, and even a single suboptimal move usually costs one the contest [Cohen, 1972]. For X there is typically exactly one correct move, including the single correct opening. As a result, the perfect play algorithm must be quite rigid and offers little opportunity to acquire power during training.

Only after perfect training was Hoyle always perfectly reliable, and then only against the perfect challenger. After perfect training Hoyle still lost 12% of its contests to the expert, 18% to the novice, and 19% to the random challenger. Clearly, learning had been incomplete. (Inspection revealed that during training Hoyle usually met the behavioral standard by the endgame skill it had acquired.) For any other training, reliability was dramatically worse; losses to the perfect challenger averaged from 40% to 61%, to the expert from 4% to 46%, to the novice from 14% to 29%, and to the random challenger from 7% to 23%. The program sporadically achieved perfect reliability against the perfect or the expert challenger on a single run for several low e values, but still went on to lose at least four testing contests against the other challengers in the same run. The most consistently reliable performance against the expert was achieved by $e = 20$, against the novice by $e = 20$ and $d = 2$, and against the random challenger by $d = 5$. Reliability against the perfect, expert, and novice challengers decreased with the error rate, i.e., the trainer's lack of skill appears to have misguided the learner. Reliability against the random challenger, however, *increased* with the error rate.

Hoyle's power at lose tic-tac-toe ranged from 14% to 41% against the expert, from 54% to 72% against the novice, and from 60% to 80% against the random challenger. Maximal power against the random challenger was developed from training with $d = 5$, against the novice with $e = 80$, and against the expert in three trials, with $e = 80$, with random training, and with $d = 4$. Power against the expert challenger and against the random challenger *increases* with the error rate of the trainer. Presumably *the lack of errors during training made the program less able to maneuver in the search space when the testers erred.*

Learning time ranged from 10 contests, during two runs for $e = 80$, to 156 for a run when $e = 70$. Learning space ranged from 56.8 units for $d = 7$ to 340.8 for $d = 3$. Learning space increased with learning time for fallible training. The self-trained program lost 61% of its contests to the perfect challenger, 46% to the expert, 23% to the novice, and 22% to the random challenger, but also managed to win 14% against the expert, 66% against the novice, and 69% against the random challenger.

4.1.3 Achi

Achi is a game where most serious errors occur early, in the stage when markers are first placed on the board. Contests average 68 moves, offering ample opportunity for careless error while play cycles to a draw.

When $e = 20$, 30, 80, 90 and when $d = 2$, 4, 5, 6, 8 perfect reliability was achieved consistently, i.e., against all the challengers in every run. For other training, losses were rare (about .03%) and never to the expert challenger. Hoyle's power in achi ranged from 58% to 77% against the expert, from 97% to 100% against the novice, and 99% to 100% against the random challenger. Maximal power against the expert was achieved with $e = 20$.

Learning time ranged from 10 to 13 contests, but was greater than 10 in only 3 runs. Learning space ranged from 1 unit, for self training, all informed training, and perfect training, to 52.4 for $e = 90$. Learning space is positively correlated with learning time for fallible training. Inspection of the useful knowledge cache revealed that, against a fallible trainer, the program always learns some accurate and quite sophisticated achi strategy that it does not acquire during perfect training. The resultant increase in its learning space from fallible training does not, however, improve the program's reliability or make a statistically significant change in its power. Although the knowledge was correct and clever, it had no visible impact on the evaluation criteria posited here, i.e., *the fallible achi trainer induced learning that was a waste of resources.* The self-trained program loses 1% of its contests to the random challenger, but also manages to win 72% against the expert, 100% against the novice, and 99% against the random challenger.

4.2 The impact of trainer error

Although it is possible to train a program to be at least fairly reliable for each of the three games, imperfect training offers better preparation for the occasional opportunities that arise across a broad range of competition. In lose tic-tac-toe, the most difficult of the three games for Hoyle to learn, the e value is correlated with the number of wins against the expert challenger; the *more* fallible the trainer, the more powerful the program. *When learning is incomplete, Hoyle's best preparation for imperfect play is a fallible trainer.* After perfect training the program was 100% reliable against the perfect challenger, but only 88% against the expert challenger, 82% against the novice challenger, and 81% against the random challenger.

When self training is compared to the entire range of e values for fallible training under the similarity metric of Section 3, there is a dramatic and distinctive similarity between self testing and $e = 60\%$ for tic-tac-toe, $e = 50\%$ and 80% for achi (where the difference from self training is almost 0), and $e = 70\%$ for lose tic-tac-toe, i.e., *self training is like learning against a fallible player many of whose moves may be errors.* At tic-tac-toe, self training produced the lowest power against the novice and near the lowest against the expert; it was also only 70% reliable against the perfect challenger. At lose tic-tac-toe, self training produced the lowest power against the expert and the lowest or near the lowest reliability against every challenger. Only at achi, the game where useful knowledge had the least impact, was self training moderately reliable and powerful.

The difference between learned behavior after informed trainer error and after random trainer error appears strongly related to both the nature of the evaluation function and the

game. For tic-tac-toe and achi, informed training relied upon the number of potential winning and losing lines (up to 8) on the board. This greedy approach always opens in the center, regardless of depth. As a result, no informed training offers Hoyle the opportunity to learn the simple fork from a corner opening that made the learner so reliable after perfect training at tic-tac-toe. Hoyle learns a minimum during informed training at any depth in these two games, so its resultant testing behavior is simply the luck of the draw against its challengers.

For lose tic-tac-toe, however, unless the trainer makes a lot of mistakes, a program that meets the behavioral standard is going to have to learn to open in the center. This is a move most people, and therefore the evaluation function we used, find highly counterintuitive. An informed trainer, regardless of depth, will never open in the center: either it will not search deeply enough or it will exercise its left-to-right bias. For Hoyle to learn the correct opening against an informed trainer, it must observe both a corner and a side opening, and then prove that those openings, rather than any later move, were responsible for the subsequent losses. Even then, the program must learn how to play *after* the correct opening. Against a fallible trainer, there will be more opportunity for this to happen; against an informed trainer only half the contests (where Hoyle goes first) even offer the opportunity to learn to play X perfectly. As a result, informed training in lose tic-tac-toe is often slower than fallible training, and the resultant quality of play can be weaker.

5. Related Work

This research differs from prior work by educators and psychologists because it is able to start each learning experience with a machine that offers a *tabula rasa,* a clean slate. Because people cannot clear their minds of all prior experience, and because they may learn differently, the results may not be analogous to people. Hoyle's learning speed and output results, however, do simulate an experienced game player encountering an unfamiliar game [Epstein, 1992].

Neurogammon, an earlier version of TD-gammon, was a neural net program that learned to play backgammon [Tesauro and Sejnowski, 1989]. The program was trained to select the moves made in 400 contests where Tesauro, a strong but not world-class player, had played both sides. At the First Computer Olympiad in London in 1989, Neurogammon was clearly the strongest non-human competitor. When Neurogammon plays TD-gammon, however, it only wins 40% of the time. There are several obvious explanations for TD-gammon's improved strength. First, TD-gammon had much more extensive training; it learned on approximately 200,000 contests. Second, TD-gammon uses the TD(λ) algorithm instead of Neurogammon's standard back-propagation [Sutton, 1988; Rumelhart *et al.*, 1986]. Finally, TD-gammon trains against itself, probably with a more varied set of experiences.

An alternative kind of training attempts to prime the learner, to provide it with a head start by first observing two perfect players in competition before making any moves itself. Experiments with a simple pattern-learner and reinforcement training for several games on a three-by-three grid have indicated that priming *slows* learning [Painter, 1992]. One possible explanation for this is that the initial bias so developed is irrelevant, or even wrong, for many of the game states that the novice learning program soon faces. The head start must thus be partially unlearned before useful learning can take place. Painter also found that a random trainer produced a less reliable player. N-N/Tree, a hybrid learning program with a neural net, was also found to suffer from priming [Flax *et al.*, 1990]. N-N/Tree also learned to play far better against a fallible player with $e = 5$ than against a perfect player.

6. Conclusions

The role of the trainer in a competitive machine learning experience has usually been a matter of convenience. Input book games require the tedious assembly of databases. Human opposition of any caliber plays too slowly, tires too quickly, and may be fairly rigid in approach. That leaves only opposition from another machine.

One way to see training for a competitive domain is as a set of paths through a game tree. If the trainer always plays perfectly, the learner will have no experience with large portions of the problem space. After such an overly narrow learning experience, there is no reason to believe that a program will have the skill to deal with errors, or even with suboptimal moves, let alone exploit them to its advantage. The data presented here confirm this, particularly in a game where relatively few moves are good choices. (Go is reputed to be such a game.) A competitive learning experience against a perfect player is flawed, and the resultant performance is disappointing. No single trainer, in any of the games, achieved maximum power against all the challengers. *Training against weak opposition is inadequate preparation for a stronger opponent, but training against strong opposition also turns out to be less adequate for a weaker opponent when learning is incomplete.*

A somewhat less than perfect training experience introduces some variety into the paths through the game tree. Whether this variety is engendered by random noise or by lack of foresight and knowledge was not significant in the three games considered here. This experiment suggests that a trainer informed by an evaluation function but hampered by lack of exhaustive search not only has the narrowness of the perfect trainer, but is also no more valuable as its depth increases. One might expect, however, that in a game with a larger branching factor the probability of making a suboptimal choice, rather than a terrible one, would decrease, so that lack of foresight and knowledge in a trainer would be less damaging to learning than random noise would be. The potential tradeoff between partial knowledge and the ways it might lead the learner astray seems worth some additional exploration.

It is possible that the results described here are a function of the learning program, i.e., Hoyle, rather than of the trainer. For example, one facet of Hoyle is its ability (but by no means proclivity) to imitate expertise it has observed in the opposition. TD-gammon, N/N-Tree, and Morph all imitate the opposition too, but with different learning methods from Hoyle's. A program that ignored the behavior

of the opposition might be less susceptible to the influence of its trainer, although one could argue that it was not particularly intelligent either. Tesauro attributes TD-gammon's ability to learn to play so well in part to the variety of training situations that were forced upon it by the non-determinism of the dice during learning; Gelfand found $e > 0$ essential [Tesauro, 1991; Flax *et al.*, 1990]. The results described here, in conjunction with theirs, suggest that the conclusion about the need for variety in training is a function of the learning task, not of the particular learning methods used here.

When learning time was relatively constant, learning space was observed (87.5% correlation for achi, 93.75% for tic-tac-toe) to increase with the fallibility of the trainer. When learning time in fallible training varied widely (in lose tic-tac-toe), learning space varied with it. In both cases, this is because the heuristics pick up a good deal of data that probably does not strengthen play, simply because the program was exposed to so many mistakes. A neural net has no such difficulty. It would be interesting to see whether other methods also have larger long-term memory requirements when training against weaker opposition.

Learning time was dependent on many factors. Hoyle simulates an expert game player; it has general heuristics for playing all games even before it learns about any specific game. This makes its initial level of play higher than, say, a neural net that randomized its initial weights. As a result, Hoyle could meet the behavioral standard relatively quickly in tic-tac-toe and achi. A fallible trainer often introduces peculiar situations that the program would fail on, but learn not to repeat, with a resultant increase in learning time. The more fallible the trainer, the more often this can happen. On the other hand, a fallible trainer also makes mistakes that the program can exploit. The more fallible the trainer, the more often this happens too, so that a highly fallible trainer may allow the program to meet the behavioral standard too quickly. All these factors visibly interacted, masking any correlation.

Self training offers a natural, gradual progression from weaker to stronger. It should therefore prepare the program for any opposition as good as itself. Of course, using a program as its own trainer deprives it of an important knowledge source that people learn from, the expert model. Self training is also a substantially slower way to acquire broad expertise; TD-gammon spent the first 25% of its training playing "long, looping contests that seemed to go nowhere" [Tesauro, 1991].

One solution to the high estimated e values similar to self training might be to raise the behavioral standard above 10 to lengthen the training time. In a game with a small search space this should augment Hoyle's useful knowledge and improve its performance, but there can be no guarantee of perfection. This was demonstrated in single runs with e from 20 to 60 for lose tic-tac-toe, with the behavioral standard set at 100 instead of 10. Learning time with this higher behavioral standard ranged from 297 to 505 contests, instead of 10 to 156, and learning space from 336 units to 809, instead of 56.8 to 340.8. In every run with the higher behavioral standard, despite the fact that Hoyle had seen much more of the search space, the program still lost from 1% to 2% of its contests and showed no significant change in power. It is important to note that those losses were only to the novice or the random challenger. *A higher behavioral standard improved reliability but not power.* A higher behavioral standard also substantially increased learning time and memory requirements. In 505 contests averaging 9 states with markers on the board, Hoyle encounters 4545 (not necessarily distinct) states out of 5478 possible distinct states in the entire search space. Even after the opportunity to encounter as much as 83% of the entire search space, Hoyle is not perfectly reliable. *What is required is not only more training but broader based training, i.e., novel experiences.*

For games without an element of chance, variety in training can be introduced with a broad spectrum of opposition. As a result of this experiment, we recommend a hybrid training experience for any program in a competitive domain, one that interleaves sessions against a perfect trainer with practice against itself. That is the method Hoyle now uses in discovery mode, to develop its own expertise without human guidance. When Hoyle trains this way, it learns to be perfectly reliable and very powerful against all the challengers.

For more difficult games where no perfect trainer is possible, this research offers no reason to believe that self-training can ever result in perfect play. For such games, these results would advocate *lesson-and-practice training:* alternating sets of a few contests against the best player available (the lesson) and many contests against the learning program itself (the practice). Whether or not a perfect player is available, training should be periodically spiced with particularly fallible opposition and a random move or two, to keep a novice from taking the learner by surprise. Under this regimen, once game-learning programs surpass people, they can continue to train against each other. To the extent that such programs develop different styles of play, competition among them should strengthen their abilities. In so imperfect an environment, however, there can be little guarantee that they will ever play perfectly. A program can meet a behavioral standard and still have much to learn.

Acknowledgments

The author thanks Jack Gelfand and Gerry Tesauro for insightful discussions. Kouros Esfahany and Joanna Lesniak provided expert-level data generation and programming support.

References

[Anantharaman *et al.*, 1990] T. Anantharaman, M. S. Campbell, and F.-h. Hsu. Singular Extensions: Adding Selectivity to Brute-Force Searching. *Artificial Intelligence* 43 (1): 99-110, 1990.

[Berliner and Ebeling, 1989] H. Berliner and C. Ebeling. Pattern Knowledge and Search: The SUPREM Architecture. *Artificial Intelligence* 38 (2): 161-198, 1989.

[Cohen, 1972] D. I. A. Cohen. The Solution of a Simple Game. *Mathematics Magazine* 45 (4): 213-216, 1972.

[Conover, 1980] W. J. Conover. *Practical Non-Parametric Statistics,* second edition. John Wiley and Sons, New York, NY, 1980.

[Epstein, 1991] S. L. Epstein. *Learning under a Weak Theory*, Technical Report 91-01, Department of Computer Science, Hunter College, 1991.

[Epstein, 1992] S. L. Epstein. Prior Knowledge Strengthens Learning to Control Search in Weak Theory Domains. *International Journal of Intelligent Systems,* to appear.

[Flax *et al.*, 1990] M. G. Flax, J. J. Gelfand, S. H. Lane, and D. A. Handelman. Integrating Neural Network and Tree Search Approaches to Produce an Auto-Supervised System that Learns to Play Games. In *Proceedings of The Aerospace Applications of Artificial Intelligence Conference,* Dayton, OH, 1990.

[Levinson and Snyder, 1991] R. Levinson and R. Snyder. Adaptive Pattern-Oriented Chess. In *Proceedings of the Eighth International Machine Learning Workshop,* 85-89. San Mateo, CA, August, 1991. Morgan Kaufmann.

[Nilsson, 1980] N. J. Nilsson. *Principles of Artificial Intelligence.* Tioga Publishing, Palo Alto, CA, 1980.

[Painter, 1992] J. Painter. Pattern Recognition for Decision Making in a Competitive Environment. Master's diss., Hunter College of the City University of New York, in preparation.

[Rumelhart *et al.,* 1986] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Internal Representation by Error Propagation. In *Parallel Distributed Processing,* Vol. 1, ed. D. E. Rumelhart and J. McClelland. MIT Press, Cambridge, MA, 1986.

[Samuel, 1963] A. L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. In *Computers and Thought,* ed. E. A. Feigenbaum and J. Feldman. McGraw-Hill, New York, NY, 1963.

[Samuel, 1967] A. L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. II - Recent Progress. *IBM Journal of Research and Development* 11 (6): 601-617, 1967.

[Sutton, 1988] R. S. Sutton. Learning to Predict by the Methods of Temporal Differences. *Machine Learning* 3 (9-44, 1988.

[Tesauro and Sejnowski, 1989] G. Tesauro and T. J. Sejnowski. A Parallel Network that Learns to Play Backgammon. *Artificial Intelligence* 39 (3): 357 - 390, 1989.

[Tesauro, 1991] G. Tesauro. Personal communication.

[Tesauro, 1992] G. Tesauro. Practical Issues in Temporal Difference Learning. *Machine Learning,* to appear.

Training Networks Of Value Units

Michael R.W. Dawson, Don P. Schopflocher,
James Kidd, and Kevin Shamanski

Biological Computation Project
Department of Psychology
University of Alberta
Edmonton, Alberta

ABSTRACT

A modification of the generalized delta rule is described that is capable of training multi-layer networks of value units -- that is, units defined by a particular nonmonotonic activation function, the Gaussian. For simple problems of pattern classification, this new rule produces networks that appear to have several advantages over standard feedforward networks.

Introduction

Physiological evidence clearly indicates that the brain is not a network of homogenous processors. "No longer can neural networks be viewed as the interconnection of many like elements by simple excitatory or inhibitory synapses. Neurons not only sum synaptic inputs but are endowed with a diverse set of intrinsic properties that allow them to generate complex activity patterns" (Getting, 1989, p. 187). Furthermore, physiological evidence also indicates that some neural properties, such as membrane potential -- one plausible physiological correlate of "unit activation" -- vary nonmonotonically with input (for example, see Hodgkin & Huxley, 1952, Fig. 6 to see nonmonotonicity of sodium conductance).

The relationships between simulated and real neural networks could be strengthened by exploring connectionist architectures that can include nonmonotonic activation functions. A processor with a simple nonmonotonic activation function would only generate strong responses if its net input fell within a particular range; if the net input is too small or too large, the processor would not respond. Such processors are called *value units* by Ballard (1986). However, value units are not usually found in parallel distributed processing (PDP) architectures. For instance, a prevailing algorithm for training PDP networks is the *generalized delta rule* (Rumelhart, Hinton, & Williams, 1986a, 1986b). This rule was derived under the assumption that the activation of processing units is a continuous, monotonic function of their net input (e.g., a function like the logistic, see Rumelhart et al, 1986b, pp. 324-325). Ballard (1986) calls a processor with this type of sigmoid-shaped activation function an *integration device*. Thus, a system trained by the generalized delta rule is a homogenous network of integration devices. Unfortunately, a network that includes value units cannot be usefully trained by the standard version of the generalized delta rule (see below).

In this paper, we describe a variation of the generalized delta rule that is capable of training feedforward neural networks that include value units. We also present the results of computer simulations which suggest that this paradigm has several advantages when compared to the standard generalized delta rule. First, the modified rule leads to faster training on a number of problems. Second, the modified rule allows us to train hybrid networks, which contain both integration devices and value units. Third, specific properties of the value unit architecture lead to networks that are easier to interpret, because they are simpler in structure (i.e., they contain fewer processing units and connections). Fourth, networks trained by the modified rule appear to generalize their performance to new instances better than standard networks.

A Learning Rule For Value Unit Networks

The generalized delta rule is a supervised learning procedure that uses the difference between a network's observed and desired output as the basis for changing the strengths of its connections. Rumelhart et al. (1986a, 1986b) define the response error of a network with n output units to some input pattern p as:

$$E_p = \frac{1}{2}\sum_{j=1}^{n} (T_{pj} - O_{pj})^2 \qquad (1)$$

In this equation, T_{pj} represents the desired response of output unit j to the pattern, and O_{pj} represents the

observed response of the unit. Rumelhart et al. derived two rules for modifying the connection weights in different parts of the network that lead to a reduction in error as defined in Equation (1). The first rule specifies how the weights of connections leading directly to output units are to be changed as a function of the response error calculated for the output units. The second rule specifies how output unit error can be used to compute the response error for processors in an adjacent layer of hidden units. Once hidden unit error is determined, hidden unit connections can be modified, and the error for the next layer of hidden units can be calculated. This process is repeated until all network connections have been modified.

The equations for the generalized delta rule were derived under the assumption that all network processors were integration devices. We now consider a different network architecture, in which every output unit is a value unit; we call this a *value unit network*. One simple expression that can be used to model the inverted-U shaped activation function of any value unit j is the Gaussian:

$$G(net_{pj}) = EXP[-\pi (net_{pj} - \mu_j)^2] \qquad (2)$$

where net_{pj} is the net input to unit j when pattern p is presented to the network, and μ_j is the "bias" of the activation function. This equation defines a normalized version of the Gaussian which achieves a maximum value of 1 when $net_{pj} = \mu_j$, and rapidly decreases towards 0 as net_{pj} is increased or decreased from this optimal value. The properties of this function are described in detail by Bracewell (1978, pp. 53-57).

In principle, a value unit network can be trained using the generalized delta rule (i.e., by replacing the term $f'_j(net_{pj})$ in Rumelhart et al.'s [1986b] Equations 13 and 14 with the first derivative of Equation 2). In practice, however, this rule is not useful when the Gaussian activation function is used -- we have found that this type of training will frequently reduce system error to a local minimum in which the network correctly asserts that some property is not true of pattern p, but *fails* to correctly assert that some property is true of pattern p.

This problem can be overcome as follows: One property of a perfectly trained value unit network is that, for any pattern p, the value of Equation (1) is 0. A second property is that $net_{pj} = \mu_j$ for any pattern p with desired output $T_{pj} = 1$. In effect, this second property provides heuristic information that can be used to "steer" the network's search through the error space such that local minima of the type described above are avoided. This heuristic information can be added to the learning algorithm by redefining system error (i.e., Equation 1) to include *both* of these properties.

Consider the following cost function C_p as a measure of the response error to some pattern p for a value unit network with n output units:

$$C_p = \frac{1}{2}\sum_{j=1}^{n} (T_{pj} - O_{pj})^2 + \frac{1}{2}\sum_{j=1}^{n} T_{pj} \cdot (net_{pj} - \mu_j)^2 \qquad (3)$$

The first component of C_p measures the failure of the system to match the observed output with the desired output, and is identical to Equation (1). The second component measures the failure of the system to set $net_{pj} = \mu_j$ when the desired output is equal to 1. As defined, this second component *requires* that T_{pj} be either 0 or 1; this limits the network to (ideally) generating binary outputs after training. By making the minimization of Equation (3) the goal of learning, the local minimum problem described above is avoided, because the second term in C_p prevents the weights being changed such that *all* of the net inputs are drawn towards infinity.

A learning rule for a system whose response errors are defined as in Equation (3) must specify how some change in weights $\Delta_p w_{ij}$ will decrease the error term C_p for any pattern p. Because the Gaussian activation function is differentiable, such a rule can be defined by deriving a term proportional to $-\delta C_p / \delta w_{ij}$ in a fashion analogous to Rumelhart et al. (1986b, pp. 325 - 326). It can be shown that the desired change in the weight of any connection terminating in some output value unit j is:

$$\Delta_p w_{ij} = \eta \cdot (\delta_{pj} - \epsilon_{pj}) \cdot I_{pi} \qquad (4)$$

Equation (4), the learning rule for a value unit network, is clearly an extension of the generalized delta rule. The δ_{pj} term in Equation (4) is equal to $(-T_{pj} - O_{pj}) \cdot G'(net_{pj})$, and is equivalent to the same term in Equation (11) of Rumelhart et al. (1986b), with the notable exception that the first derivative is not of the logistic function, but is instead of the Gaussian. The ϵ_{pj} term in Equation (4) is equal to $T_{pj} \cdot (net_{pj} - \mu_j)$, and is the result of augmenting the to-be-minimized error function (i.e., the result of converting E_p to C_p). It serves to meet the condition that $net_{pj} - \mu_j = 0$ when $T_{pj} = 1$.

The error term calculated for some output value unit j (i.e., the term $\delta_{pj} - \epsilon_{pj}$) can be propagated backwards to layers of hidden units as is done with the standard generalized delta rule (see Rumelhart et al. [1986b] Equation 14). Such hidden units can be either

value units or integration devices. Error is defined for a hidden value unit with the Gaussian's derivative:

$$\delta_{pi} = G'(net_{pi}) \cdot \sum_{j=1}^{n} w_{ij} \cdot (\delta_{pj} - \epsilon_{pj}) \qquad (5)$$

The error for a hidden integration device is similarly defined using the first derivative of the logistic equation:

$$\delta_{pi} = f'(net_{pi}) \cdot \sum_{j=1}^{n} w_{ij} \cdot (\delta_{pj} - \epsilon_{pj}) \qquad (6)$$

Results Of Computer Simulations

To explore the utility of the learning rule for value unit networks, we used the following procedure: The starting state for a value unit network was determined by setting all connection strengths and all values of μ_j to random numbers selected from the range [-1, 1]. Learning then proceeded by running the network through a series of training epochs, during which the network was presented each to-be-learned pattern in a random order. A stochastic gradient approach (e.g, Widrow & Stearns, 1985) was adopted -- that is, network components were updated using the equations described above after *each* pattern presentation. The learning rate η was equal to 0.1 unless otherwise noted. Momentum was not used to speed learning, although a momentum term can be successfully incorporated into this learning algorithm. Training proceeded until a correct solution was achieved -- that is, until the network generated a "hit" for all of its output units for every presented pattern. A "hit" occurred if O_{pj} was greater than 0.95 when $T_{pj} = 1$, or if O_{pj} was less than 0.05 when $T_{pj} = 0$. This definition of "hit" allowed the target values T_{pj} to be equal to 1 or 0; they were not equal to 0.9 and 0.1 as is often the case for networks of integration devices (e.g., Rumelhart et al., 1986b, p. 329). A network was repeatedly trained on a problem until 100 correct solutions were obtained in order to determine the median speed of training. If a correct solution had not been achieved after 30,000 training epochs, the training run was stopped and tabulated as a local minimum. Local minima were not included when median network performance (e.g., speed to learn) was computed.

In order to relate the performance of the value unit networks to the performance of 'standard' networks, networks of integration devices were also trained. In general, the standard procedures (e.g., Rumelhart et al., 1986a, 1986b) for the generalized delta rule were applied: network connections and bias values were started randomly in the range [-0.3, 0.3], and training proceeded with a learning rate of 0.5 and a momentum rate of 0.9 (unless otherwise stated). The only departures from standard training procedures were that a stochastic gradient approach was adopted (i.e., system error was not accrued during an epoch), the target output values T_{pj} were either 0 or 1, and the definitions of "hit" and "correct solution" used to determine the convergence of a value unit network were also applied here.

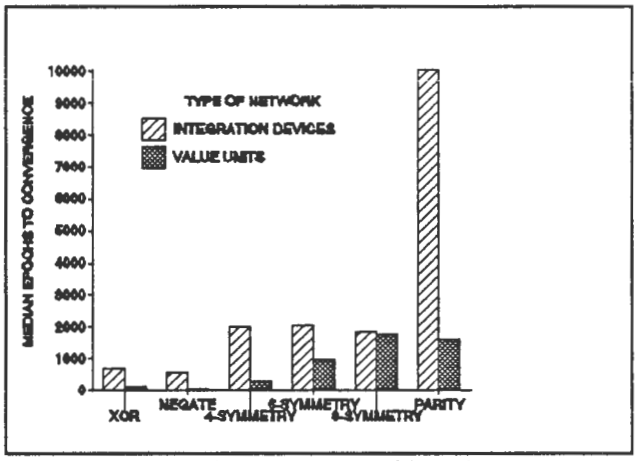


Figure 1 Comparison of speed of learning in value unit networks and networks of integration devices. Histograms indicate median epochs to learn over a set of 100 different training sessions for each architecture.

Speed of Training

The first set of simulations compared the number of epochs required to train the two types of networks. We used a number of the "toy problems" studied by Rumelhart et al. (1986b). The median numbers of epochs required to reach convergence for some of these problems are illustrated in Figure 1; more details are provided in Dawson and Schopflocher (1992). In general the value unit networks learned the solutions to these problems faster than did the standard networks. One-tailed dependent t-tests indicated that this difference was statistically significant ($t = 1.928$, $df = 11$, $p < 0.05$). Fewer local minima were encountered by the value unit networks as well, although for these problems local minima were rare for both architectures (14 totalling across problems for the value unit networks, 75 for the standard networks). This difference was also statistically significant ($t = 1.778$, $df = 11$, $p < 0.05$). Nevertheless, these results do not indicate that the value unit network is a "better" or "more practical" architecture in general. This is because there are many differences between the two types of networks that exist as free parameters that could be varied in such a way that a closer match in performance might be obtained. The key point underlying the statistical

tests is that when the generalized delta rule, defined with typical settings, is used as a benchmark (c.f., Barnard & Casasent, 1989), the performance of the value unit rule is more than satisfactory.

Training Hybrid Networks

A second set of simulations was performed to examine the ability of the modified learning rule to train hybrid networks, which consisted of an output layer of value units, and a hidden layer of integration devices. Our major interest in training such networks was to take a small step towards increasing the biological plausibility of PDP networks; real neural networks are not comprised of homogenous processors (e.g., Getting, 1989). Importantly, the ability to train such networks also permits a more appropriate comparison of learning speed than in the simulations above, because greater control can be achieved over artifactual differences between the two architectures.

The logic of this more rigorous comparison is quite straightforward: in a control condition, a homogenous network is trained to perform a task, and its speed to learn is measured. A test condition is then created by replacing parts of the control network with processors from the other architecture. This test network is trained, and its speed to learn is also measured. In this design, the basic patterns of connectivity (i.e., the number

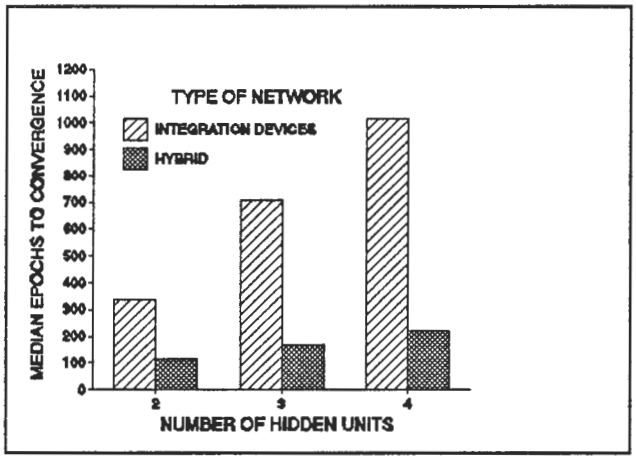


Figure 2 Comparison of learning speeds for homogenous network of integration devices vs. a hybrid network with value units in the output layer and integration devices in the hidden layer.

of processing units and the number of initial connections) for the control and the test networks are identical. Furthermore, during learning the common components of the two networks are treated identically -- they are

In one of our experiments, the control system was a homogenous network of integration devices trained with a learning rate of 0.5 and a momentum of 0.9. The test system was a hybrid network in which the outputs were value units; the hidden units were still integration devices. A learning rate of 0.025 was used to train the μ_j parameters and the weights of the direct connections to the output units. All other aspects of the test system were the same as the control -- all connection weights and unit biases (including those of the value units) were selected randomly from the range [-0.3, 0.3], all connections and biases were modified using a momentum term of 0.9, and the learning rate for the hidden unit biases and connections was 0.5 as in the control system. Each network was given 100 different training sessions on each problem, and the median speed to learn was computed.

The results of this experiment are presented in Figure 2. Replacing the outputs of a homogenous network of integration devices with value units resulted in a dramatic improvement in speed of processing for all three problems. These results raise the intriguing possibility that the performance of existing "standard" backpropogation networks could be improved by replacing their output integration devices with value units and by training the resulting hybrid network with the modified generalized delta rule.

Why does a network with output value units learn the encoder problem so much faster than does a network with output integration devices, even when the internal components of the two networks are identical? Learning in a feedforward network can be described as a heuristic search through a parameter space (e.g., Sandon & Uhr, 1988). From this perspective, Equations (1) and (3) can be viewed as evaluation functions that are being minimized by a gradient descent procedure, and can be compared in terms of their heuristic power (e.g., Nilsson, 1980, pp. 79-81). Clearly Equation (3) contains more information about the space being searched because it contains Equation (1) as its first component. The second component of Equation (3) introduces additional heuristic information about the Gaussian activation function that leads to more efficient search. Note that the end product of either gradient descent is the same, for it can be shown that the minimum values of Equations (1) and (3) are identical.

Value Unit Networks And Interpretation

Mozer and Smolensky (1989, p. 3) have noted that "one thing that connectionist networks have in common with brains is that if you open them up and peer inside, all you can see is a big pile of goo". This poses a major problem for connectionist research, because a trained network can be construed as an explanation of some phenomenon, and thus should be interpretable. In

response to this problem, some PDP researchers are currently concerned with discovering methods to reduce network size in order to increase network interpretability or performance (e.g., Hagiwara, 1990; Mozer & Smolensky, 1989; Sietsma & Dow, 1988). With respect to this issue, two properties of the value unit architecture lead to smaller networks -- fewer processors and connections -- and thus may lead to more interpretable systems.

First, because it possesses a nonmonotonic activation function, a single value unit is not restricted to making linearly separable discriminations (see Dawson & Schopflocher, 1992, Figure 2). Thus, in general, value unit networks should require fewer processors to make the same discrimination than an integration device network. Indeed, a single value unit can perform the same function as a circuit in which one integration device receives input from two others, as is illustrated in Figure 3 for the XOR problem.

A second potential advantage of value unit networks was revealed as a byproduct of some research in which we have attempted to place stronger biological constraints on one aspect of the architecture, the parameters μ_j. Electrophysiological studies of neurons have shown that an action potential will be generated when the membrane potential exceeds a threshold voltage (the "all-or-none law", see Kandel, 1991, Figure 2-8). This threshold voltage is a likely physiological correlate of a PDP unit's bias (i.e., the term θj for integration devices, or the term μj for value units). Using standard training procedures for either architecture, unit biases are not constant, but are instead modified during learning along with network connections. However, in neurons, the threshold membrane potential appears to be fixed -- it is *not* modified by learning. As a result, we have begun to explore the effects of training networks in which bias is a fixed parameter.

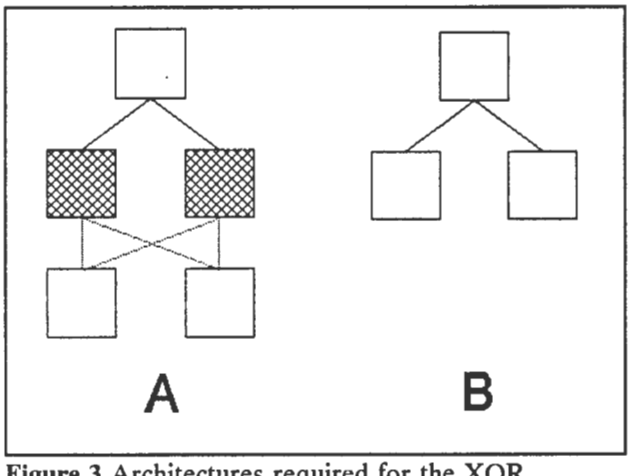


Figure 3 Architectures required for the XOR problem. (a) A standard network of integration devices. The crosshatched boxes and broken lines represents additional units and connections required in comparison to (b), a value unit architecture.

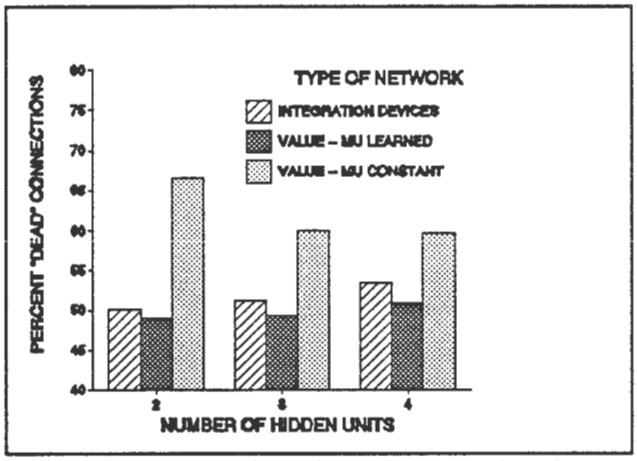


Figure 4 The proportion of "dead" connections in a value unit network increases when bias terms are fixed throughout learning.

In one study, two control conditions were used: a homogenous network of integration devices, and a homogenous network of value units. Both of these control networks were trained using the standard procedures described previously. The experimental condition was a homogenous network of value units in which each μ_j was fixed at 0 throughout training. We compared the performance of these networks on three different versions of the encoder problem (4-2-4, 8-3-8, 16-4-16). One dependent measure was the percentage of "dead connections" at convergence -- the proportion of connection weights whose absolute value was less than 0.01. As can be seen in Figure 4, for both control networks roughly 50% of the possible connections were "dead" at trainings end. In contrast, when value unit bias was held constant, the percentage of "dead" connections was raised to 60% or better. This increase in network simplicity was statistically significant -- for each version of the encoder problem, t-tests indicated that the constant bias networks had significantly more dead connections than did the standard value unit networks (t = *13.662, 12.231, 13.772* for the three problems respectively, df = *99*, $p < .01$) and the standard integration device networks (t = *28.034, 15.554, 9.271*, df = *99*, $p < .01$). Interestingly, we have as yet been unable to train a single network of integration devices to convergence on an encoder problem when bias is fixed, even with exploring a wide range of learning and momentum rates, with a variety of starting configurations, and with liberalizing our definition of a local minimum to a failure to converge after 1,000,000 epochs.

Performance On A Larger Problem

The preceding computer simulations demonstrate certain potential advantages of the value unit architecture over networks of integration devices. However, these simulations have only examined very small problems. One reason for the popularity of the generalized delta rule in cognitive science has been its ability to train large networks on difficult and interesting problems (e.g., Seidenberg & McClelland, 1989). Furthermore, in many of these problems networks of integration devices with n input units and m output units are taught mappings from $\Re^n$ to $\Re^m$. The preceding simulations have only examined Boolean mappings from $\{0,1\}^n$ to $\{0,1\}^m$. Our final set of results arises from part of a research programme designed to answer three different questions about value unit networks: Can a value unit network learn a mapping when input activations are not binary? How well does the learning rule train larger networks? How well does performance generalize to novel instances?

The second component of Equation (3) requires a value unit network to generate binary outputs. As a result, we were interested in determining whether we could train such a network a mapping from $\Re^n$ to $\{0,1\}^m$ for a complex problem. We decided to train a network to identify membership in the Mandelbrot set, which is well known in the literature on chaotic dynamical systems (e.g., Devaney, 1989, pp. 311-319). In essence, the Mandelbrot set is defined in a two-dimensional plane; one dimension corresponds to real numbers, the other dimension corresponds to complex numbers. For any point in this plane, a simple iterative function can be computed to determine if the point is in the Mandelbrot set. If the value of this function rapidly increases (towards infinity) with repeated iterations, it is said that the point is not in the set; otherwise, the point is said to be part of the set.

We created one version of the Mandelbrot set using an algorithm described by Peitgen (1988, pp. 190ff). The figure that we created was a sample of 2500 equally spaced coordinates (a 50 by 50 display), ranging from -2.0 to 0.6 on the real axis, and whose real component on the imaginary axis ranged from -1.2 to 1.2. Membership in the Mandelbrot set was determined after 100 applications of the iterative equation. Two networks, one composed of value units, the other of integration devices, were trained on 500 coordinates from this display, randomly selected apart from the restriction that half of the sampled coordinates were in the set, and half were not. The value unit network had two input units that registered the coordinates of a point in the display, two layers of hidden value units with 20 units per layer, and a single value unit for output. The network of integration devices had a similar structure, with the exception that there were 40 units in each layer of hidden processors. Each network was trained on the 500 instances for a specific number of epochs (ranging from 100 to 1000), and was then presented all 2500 points from the display to determine how well learning generalized to novel instances. Each of these training sessions was run 10 different times to examine average performance from different random starts.

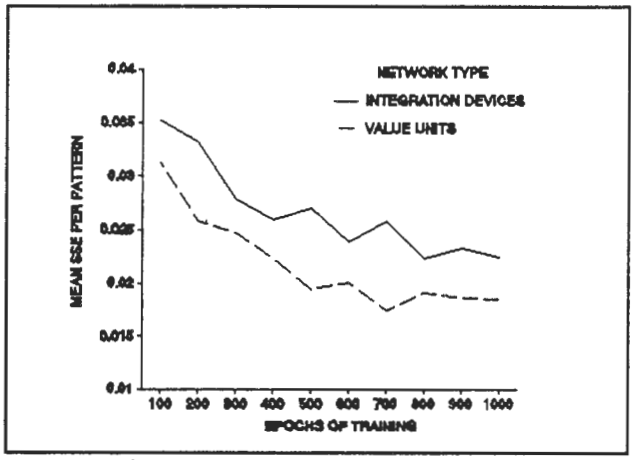


Figure 5 The average sum of squared error per pattern for two networks trained on the Mandelbrot set. These averages are taken over 2500 pattern instances, and 10 separate training runs.

The resulting data indicated that both networks did quite well at learning the coarse structure of the stimulus from the presented instances; this learning appeared to generalize nicely to points that had not been presented. In addition, the data suggested that the value unit network's performance was better than the network of integration devices; finer details of the Mandelbrot set seemed to appear earlier. A more objective test of this was performed by computing the sum of squared error of each network's output. These results are illustrated in Figure 5, which shows that the value unit network's response was more accurate than was the network of integration devices throughout training. Thus, with respect to our three general questions, the modified learning rule can usefully larger networks of value units; this training can incorporate continuous input values; this training also appears to generalize to novel instances. Furthermore, the advantages of the value unit architecture on the "toy problems" described earlier persist when more complicated mappings are presented to a substantially larger network.

References

Ballard, D. (1986). Cortical structures and parallel processing: structure and function. *The Behavioural and Brain Sciences, 9*, 67-120.

Barnard, E., & Casasent, D. (1989). A comparison between criterion functions for linear classifiers, with an application to neural nets. *IEEE*

Transactions on Systems, Man, and Cybernetics, 19, 1030-1041.

Bracewell, R.N. (1978). *The Fourier transform and its applications, 2nd edition*. Tokyo, McGraw-Hill Kogakusha.

Dawson, M.R.W., & Schopflocher, D.P. (1992). Modifying the generalized delta rule to train networks of nonmonotonic processors for pattern classification. *Connection Science*, accepted for publication.

Devaney, R.L. (1989). *An introduction to chaotic dynamical systems, second edition*. New York: Addison-Wesley

Getting, P.A. (1989). Emerging principles governing the operation of neural networks. *Annual Review of Neuroscience, 12*, 185-204.

Hagiwara, M. (1990). Novel backpropogation algorithm for reduction of hidden units and acceleration of convergence using artificial search. *Proceedings of the IEEE Joint International Conference On Neural Networks, Vol. I*, 625-630.

Hodgkin, A.L., & Huxley, A.F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology, 117*, 500-544.

Kandel, E.R. (1991). Nerve cells and behaviour. In E.R. Kandel, J.H. Schwartz, & T.M. Jessell (Eds.) *Principles of neural science, third edition*. New York: Elsevier.

Mozer, M.C., & Smolensky, P. (1989). Using relevance to reduce network size automatically. *Connection Science, 1*, 3-16.

Nilsson, N.J. (1980). *Principles of artificial intelligence*. Los Altos, CA: Morgan Kaufman Publishers.

Peitgen, H.-0. (1988). Fantastic deterministic fractals. In H.-O. Peitgen & D. Saupe (eds.) *The science of fractal images*. New York: Springer.

Rumelhart, D.E, Hinton, G.E., & Williams, R.J. (1986a). Learning internal errors by error propagation. In D. Rumelhart, J. McClelland and the PDP Group (Eds.) *Parallel distributed processing, Vol. 1*. Cambridge, MA: MIT Press.

Rumelhart, D.E, Hinton, G.E., & Williams, R.J. (1986b). Learning representations by back-propagating errors. *Nature, 323*, 533-536.

Sandon, P.A., & Uhr, L.M. (1988). A local interaction heuristic for adaptive networks. *Proceedings of the IEEE Joint International Conference On Neural Networks, Vol. I*, 317-324

Seidenberg, M., & McClelland, J. (1989). A distributed, developmental model of word recognition and naming. *Psychological Review, 96*, 523-568.

Sietsma, J., & Dow, R.J.F. (1988). Neural net pruning -- why and how. *Proceedings of the IEEE Joint International Conference On Neural Networks, Vol. I*, 325-333.

Widrow & Stearns (1985). *Adaptive signal processing*. Englewood Cliffs, NJ: Prentice Hall.

Acknowledgements

This research was supported by NSERC Operating Grant 2038 and by NSERC Equipment Grant 46584, both awarded to MRWD. The results on speed of training, and on training hybrid networks, are reported in Dawson and Schopflocher (1992). The other results reported in this paper have not been previously published. Thanks to Ngaire Nevin-Meadows, who conducted many of the simulations. Address correspondence to Dr. Michael Dawson, Biological Computation Project, Department of Psychology, University of Alberta, Edmonton AB CANADA T6G 2E9. Electronic mail: mike@psych.ualberta.ca

Relevancy Knowledge In Analogical Reasoning

Ye Huang and Alison E. Adam
Department of Computation
University of Manchester Institute of Science and Technology
P.O.Box 88, Manchester, M60 1QD, U.K.
huang@sna.co.umist.ac.uk
alison@sna.co.umist.ac.uk

Abstract

Analogical reasoning (AR) is a process of extending similarities. In order to justify such a process, knowledge must be employed, either implicitly or explicitly. Therefore, AR is knowledge based. This paper proposes a new form of knowledge, the relevancy rule, to support AR. It contains not only determination information, but also relevancy information. The determination part says some pieces of information are sufficient to decide some others. The relevancy part says certain information has to be there, i.e. is necessary, for deciding some other information. The relevancy rule groups all the relevant predicates together for a particular target predicate, so that this group of relevant predicates will be able to determine the target predicate and contains no irrelevant [1] information. The necessity requirement of the relevancy rule is emphasized to justify this form of knowledge, and this stands in contrast to Russell's determination rule. An AR system has been built where the learning of relevancy rules forms an important part. An inductive method of learning relevancy rules is addressed in this paper. An example of a small block world problem is used to demonstrate this approach.

1 Introduction

AR has become an active research topic within artificial intelligence and cognitive science over the past ten years. Kedar-Cabelli [1988a] gives a review from the AI perspective. Vosniadou and Ortony [1989] describe work from the cognitive approach.

Generally speaking, AR is similarity based. Its underlying assumption is that similarity in some aspects of the problem case implies similarity in some others. Since this is not true in all situations, restrictions must be introduced on the similarity extending process. That is, similarity in some aspects may only imply the similarity of certain other aspects. Therefore, knowledge about the relationships between these aspects is important. This suggests that the justification of an AR argument depends upon the underlying knowledge employed in the AR processes.

The underlying knowledge used in AR has been studied from many directions. In abstraction based AR [Greiner 1988a;1988b], knowledge is expressed as a set of abstract formulae. It is these formulae that guide the AR processes. The purpose directed AR model of Kedar-Cabelli [1988b] employs a set of purpose structures. These structures help to explain why an AR argument is made. The determination rule is a general form which represents the determination relationships between properties. In the determination base AR model [Russell, 1988], AR becomes a justifiable process.

These pioneering works guide the direction of the research described in this paper. The general understanding is that there is a body of knowledge used by AR which is different from that used in a conventional deduction system. This body of knowledge is in a weaker form than the commonly used production rule formalism [Russell, 1988] and at an abstract level that is applicable to multiple domains [Greiner, 1988a; 1988b].

This paper proposes a new form of knowledge, the relevancy rule. In some sense it is a descendant of the determination rule. However, the determination rule only expresses sufficiency requirements. $P \succ Q$ means that for any two cases, similarity in P implies similarity in Q, where P and Q are predicates. This is a general form of knowledge supporting justifiable AR. One problem with the determination rule is that there may be irrelevant properties in P. This irrelevant information may interfere with the comparison between a target case and a source case. The effect of this is that the generality of each rule is reduced. The flexibility of the AR system in dealing with new cases is thereby decreased.

The relevancy rule, as defined in this paper, has several advantages over its predecessor, the determination rule. It conveys not only determination information, but also relevancy information between properties in the world concerned. A condition which restricts applicable cases enables the relevancy rule to be case dependent, and this is an important feature of analogy.

This paper proceeds as follows: In section 2, the definition of one form of the relevancy rule is given. The discussion is focused on the necessity requirement and case restriction aspects. Section 3 concentrates on the

[1] Irrelevancy is a relative concept. Here it means every element in the group is not ignorable.

inductive acquisition of relevancy rules. The learning algorithm is outlined in this section. An example of a small block world problem is used to demonstrate this approach.

2 Relevancy Knowledge

By relevancy knowledge we mean that knowledge which conveys relevant information between properties. What exactly is relevancy then? The following are some hints for defining it. Let P and Q represent properties of a problem domain,

- When we say P is relevant to Q, we are implicitly referring to a set of other properties, A_{other}. P is relevant to Q means that there exist certain circumstances, if we keep values of the properties in the A_{other} fixed, where a change only in the value of P will be accompanied by a change in the value of Q. In other words, in order to decide the value of Q, one must inspect property P. P cannot be ignored under such circumstances.
- If we say P is relevant to Q, we assert that P will be able to change when other properties in the A_{other} are fixed, at least in some circumstances. P can only be said to be relevant to Q with respect to A_{other}.
- If we want to decide the value of Q, all relevant properties must be considered.

These suggestions imply that the relevancy relationship can be seen as one between a set of properties, $\{P_i\}$ and a single one, Q. There exist situations where if any one property in the set $\{P_i\}$ changes its value, the single property Q will also change and if none of the properties in the set changes its value, the single one will remain unchanged. This set of properties will be called the relevant set of a single property.

There are two ways of representing this relevant set, i.e. intensional and extensional representations. The intensional representation describes the relevant set implicitly by giving some assertions that all members of this set satisfy. The extensional representation gives the relevant set explicitly by listing all the elements of it. In this paper, only the extensional representation will be discussed.

2.1 Relevancy Rule: Form and Meaning

AR takes place in a world which is composed of a set of cases, which are described by a set of predicates. The term predicate is used here in a wider sense than its normal use in that its image may contain other values besides **true** and **false**. A case, for instance, can be a layout of blocks in a block world or a particular animal in an animal world (for an animal taxonomy problem). Analogy is a relationship between two cases. The AR problem is viewed as finding the proper value of a particular predicate for a particular case. This predicate is known as the *problem*, and this particular case the target case.

Let $S = \{ S_i \}$ be a set of cases and $D = \{ P_i \}$ be a set of predicates. A predicate is defined as a classification of cases. i.e.,

$P_i = \{ U_{ij}, j = 1, n\}$, where U_{ij} is a set of cases which, when the P_i is applied, will result the same value, p_{i_j}.

We require that

$$(1)\quad U_{ij} \subset S, \qquad j = 1, n;$$
$$(2)\quad U_{ij} \cap_{j \neq k} U_{ik} = \{\};$$
$$(3)\quad \cup_{j=1}^{n} U_{ij} = S.$$

The relevancy rule is a relationship defined on D for a concept $Q = P_s$. We say

$$(Condition, \{P_1, P_2, .., P_m\} \quad RR \quad Q)$$

iff

Condition $\Rightarrow T1$: $(\forall\, l)(\ (U_l \in (P_1 \otimes P_2 \otimes .. \otimes P_m))$
$\Rightarrow ((\ \exists\, j)(\ U_{sj} \in Q)$ and $(U_l \subset U_{sj}))\ )$

and

$T2$: P_i $(i = 1, m)$ are all relevant to Q with respect to $\{P_j \mid (j \neq i) and (j \in [1, m])\}$.

RR stands for relevancy relationship. $\otimes$ is a classifier operator such that

$$P_i \otimes P_j = \{U_k \mid (\exists m, n) U_k = (U_{im} \cap U_{jn})\}$$

Basically, this definition says that $\{ P_j\}$, $j = 1, m$, contains enough unignorable (necessary) information for deciding Q under the 'Condition'. The set of P_j, $j = 1, m$, is called the relevant set of Q. Clearly, there can be more than one relevancy rule for a given Q.

Term $T1$ is a sufficiency requirement; it says that the values of P_j, $j = 1, m$, determine the value of Q. Without this requirement, a comparison between P_j in two cases will be useless for deciding Q. This is basically Russell's idea of determination based AR. It is an important feature of the knowledge supporting analogical reasoning, of which Russell [1988] contains a thorough discussion. In the following paragraphs, another important feature, the necessity feature of relevancy knowledge will be discussed in depth.

2.2 Necessity Requirement of the Relevancy Rule

Term $T2$ is a necessity requirement. It says that any changes in P_j $j = 1, m$, will imply a change in Q. The reason for this is simple. If changes in P_j have no effect on Q in whatever situation, then it is irrelevant.

2.2.1 Generality

Generally speaking, the bigger the relevant set is, the more specific the relevancy rule is. In the relevancy rule guided AR model, it is preferable to choose the most general form of relevancy. Let us examine a simple case described by the following table

P_1	P_2	Q	*cases*
0	0	0	$case_1$
1	1	1	$case_2$
0	1	0	$case_3$

*Table*1

We can see that without $T2$, ($Null$, {P_1,P_2} RR Q) and ($Null$, {P_1} RR Q) are true for the cases above.

If we choose (*Null*, {P_1,P_2} RR Q), when a new case, with $P_1 = 1$ and $P_2 = 0$, is examined, it will be difficult to decide the value of Q. (*Null*, {P_1,P_2} RR Q) is too specific, and thus not flexible enough for dealing with new case.

The problem with choosing a more general form of relevancy is that it is usually more error prone than a more specific one. For example, (*Null*, {P_1} RR Q) is more error prone than (*Null*, {P_1,P_2} RR Q). In the above example, when a new case (1 0 0) is examined, (*Null*, {P_1} RR Q) will become invalid, while the more specific rule will still hold. This problem can be solved by incremental learning. i.e., the system should always be able to review its knowledge in light of new cases.

Property 1: $T2$ ensures that if (*Cond*, P RR Q), where P is a set of predicates and P' is a subset of P, then (*Cond*, P' RR Q) will not hold.

2.2.2 Redundancy

Introducing $T2$ actually helps us eliminate redundancy in the relevant set. It is easy to see that redundancy can be harmful to the comparison and mapping processes. Comparing irrelevant properties will interfere with the comparison of relevant information. If the $T2$ requirement is removed, then (*Cond*,{P_i} RR Q) will imply (*Cond*,{P_i} $\cup$ {P_j} RR Q). That is, one will be able to add anything to the relevant set. This is of course not in the spirit of relevancy and is a shortcoming of determination based AR [Russell. 1988]. Generally, eliminating redundancy will enhance the flexibility of the model in handling new cases.

Property 2: $T2$ ensures that if (*Cond*, P RR Q), where P is a set of predicates and P is a subset of P', then (*Cond*, P' RR Q) will not hold.

The redundancy discussed above is called global redundancy. Besides the global redundancy which is prohibited by the $T2$ requirement, there is another kind of redundancy, which we will refer to as local redundancy, which might possibly exist in a relevancy rule. This kind of redundancy becomes explicit if we partition all the cases (examples) into groups and split the relevancy rule for the whole set of cases into several smaller rules for each group. These new relevancy rules may contain redundancies, but they are local to the original relevancy rule. This local redundancy can be handled by the splitting process.

2.3 The Condition Term and The Splitting of Relevancy Rules

Having a condition term in the relevancy rule enables us to restrict the cases to which this relevancy rule applies. Normally relevancy is case and problem dependent. Introducing the condition term has several advantages.

Firstly, it helps the model tackle undeterministic situations where there are two or more Q values corresponding to the same set of P_i values. Given a set of cases, there may be no relevancy rule which holds for the whole set of cases, but one may exist for a subset of these cases.

Secondly, the partitioning of all cases into groups by the introduction of the condition term enhances efficiency and reduces redundancy within the relevancy knowledge.

Suppose we have a relevancy rule, rk = (*Cond*, {P_1,..,P_n} RR Q) By partitioning the cases, thus splitting the rk, we can have a set of relevancy rules, rk_i = (*Cond'*, {P'_1,..,P'_s} RR Q) where s is smaller than n. The new condition, *Cond'*, is the conjunction of the old *Cond* and a (P_i op a_i) term. P_i belongs to the old relevant set and a_i is a P_i value. This new relevancy knowledge is applicable only to the cases which satisfy the new condition.

Since the difference between n and s, $n - s$, is normally bigger than 1, introducing these new relevancy rules eliminates some local redundancy and enhances the potential efficiency and ability of the AR reasoning processes (i.e. the comparison process and mapping process).

Consider the following cases in *Table*1

P_1	P_2	P_3	Q	*cases*	P_1	P_2	P_3	Q
0	0	1	q_a	$case_1$	0	0	0	0
0	1	1	q_b	$case_2$	0	0	1	0
1	1	1	q_c	$case_3$	0	1	0	0
1	1	0	q_d	$case_4$	1	1	1	1

*Table*2 *Table*3

If we allow no other condition except '*Null*' to appear in the condition term, then only (*Null*, {P_1,P_2,P_3} RR Q) holds. Now, if a new case appears with $P_1 = 0$, $P_2 = 1$ and $P_3 = 0$, we will have difficulty in using the rule (*Null*, {P_1,P_2,P_3} RR Q). If we could produce a relevancy rule like (P_1=0, {P_2} RR Q), then it is very easy to decide the Q value for this new case.

It should be noticed that this flexibility is gained at the expense of the ability to generalize rules only from a smaller, restricted set of cases. Sometimes, there may not be enough evidence for a relevancy rule to be constructed from a small group of cases. Thus the splitting process will normally stop before the relevant set becomes empty. We can have a criterion for saying whether or not we have too few example cases. *Table* 3 illustrates this idea. (*Null*, {P_1} RR Q) is supported by all of the four examples, which represent half of all possible cases, so we can say that it is 50 percent confirmed. But if we split it into (P_1=0, *Null* Q) and (P_1=1, *Null* Q), the first one is 75 percent confirmed, while the second one is only 25 percent confirmed. If we drop the second rule, we will not be able to deal with new cases like (1 0 1), (1 1 0) and (1 0 0), which can be handled only if we do not split the (*Null*, {P_1} RR Q). Huang [1991c] discusses this further in conjunction with the measurement of inductive reliability.

Property 3: Let s be the number of predicates in the description properties set,D. Let N_c be the number of predicates in the condition and n be the number of predicates in the relevant set. If $s = N_c + n$, then (*Cond*, {P} RR Q) says nothing more than the set of example

cases in an AR paradigm where similarity is defined by identity [Huang, 1991a]. That is, this rule will not help the system to deal with new cases.

In other words, splitting a bigger rule into small ones enables generalization. It is the generalization that introduces new knowledge into the system. For example, in table 2, ($Null$, $\{P_1,P_2,P_3\}$ RR Q) gives us nothing more than the four cases listed. But (P_1=0, $\{P_2\}$ RR Q) says more than the first two cases, and (P_1=0, $\{P_3\}$ RR Q) says more than the last two cases.

3 Inductive Acquisition of Relevancy Rules

The relevancy rule based AR system is based upon the availability of relevancy knowledge, therefore, the acquisition of the relevancy rule becomes a critical problem. Conventionally, there are three ways of acquiring knowledge. The first one is to ask the user to supply the system with a set of relevancy rules. This is an efficient way if there are domain experts available. A good human computer interface and a method for eliciting relevancy knowledge can be a great help but is not within the scope of this paper. The second method is logical deduction. Once the logical properties of the knowledge are found and expressed in the form of implication rules, the deduction of new rules from known ones becomes possible [Huang, 1991b]. The third way is the method of acquiring rules by induction. Given a set of experiences, an inductive system will produce a set of rules. These rules should satisfy certain criteria. In the following subsections, only the inductive method will be addressed.

3.1 Inductive Learning

Traditionally, inductive learning refers to the process of generating descriptions or forming regularity rules from a set of data or facts. These generated descriptions or rules should account for the original data. Inductive methods have a correlational nature. Basically, they examine some facts or examples and form concepts or rules by computing the commonality and difference among these examples.

The method described here is learning by searching and evaluating. Given a set of properties, D, and a target Q, the problem is to find those P, which are subsets of D, and the proper corresponding conditions so that ($Cond, P$ RR Q) hold. That is it satisfies the definition of the relevancy relationship. In the inductive learning paradigm, the decision as to whether a relevancy rule is satisfactory or not is based upon its empirical evaluation.

3.2 Searching Strategies

The search proceeds through a space of all the subsets of D. This space has a lattice structure. The biggest element of this structure is the description set, D. The descendants of a node $\{P_i\}$ are the subsets resulting from taking one of the P_i out of this node. At each node, condition can be added to improve its evaluation.[2] Searching is guided by the following strategies.

- Searching starts from the description node, D.
- Selection is based upon the evaluation of each node. The evaluation reflects the requirements of an acceptable relevancy rule.
- If a node satisfies the requirements, then there is no need to search its descendant.
- At each level of the lattice, only the best [3] node is selected, all others will be abandoned.
- Only the immediate descendants of the best node are generated for further evaluation.
- The generation of condition term comes into play if the best node is not acceptable because of unsatisfactory T1 or IR evaluations.
- Condition term of a node is generated by a splitting process. The condition will partition the whole case space thus restrict the case space to which this node applies.
- The condition is chosen so that the resulting rules cover the maximum amount of cases.

3.3 Empirical Measures

In real world, the logical requirements described in section 2 can hardly be met. Normally, P will not exactly decide Q. In order to generate a relevancy rule from a set of real world cases, three empirical measures are employed in the evaluation of a candidate relevancy rule, which correspond to $T1$, $T2$ and inductive reliability respectively.

For $T1$, an entropy function is used to measure the homogeneity of Q value,

$$\begin{aligned} T1_{cases}(rk) &= r_{cases}(P,Q) \\ &= \sum_i prob(group_i) \times entropy(group_i) \end{aligned}$$

where $entropy(group_i) = -\sum_j prob(q_j) \times log(prob(q_j))$ and $group_i$ is the i^{th} group if we partition all example cases according to their values on the P. The prob($group_i$) is the proportion of $group_i$ against all examples. This is exactly the same as the proportional method in Quinlan's work. [Quinlan, 1983]

If $T1_{cases}(rk) < \varepsilon$, we say that this rk satisfies the $T1$ requirement. The ε is a threshold value for $T1_{cases}(rk)$ to be satisfactory.

The measure of $T2$ is defined by using the $T1$ measure. if $r_{case}(P,Q) \leq \varepsilon$

$$T2_{cases}(rk) = Max_k\{\varepsilon - r_{cases}(P - \{P_k\}, Q)\}$$

otherwise $T2_{cases}(rk)$ is undefined or meaningless.

For a relevancy rule, rk, to satisfy the $T2$, we require that $T2_{cases}(rk) < 0$.

The inductive reliability (IR) measure concerns the proportion of the training examples to the overall space of cases. Generally, more evidence will lead to a stronger belief in the generated rule, thus a higher IR. But the

[2] In fact, the condition part of the relevancy rule forms a more complicate structure.

[3] Heuristics, such as restricted maximum generalization, are used.

IR is also dependent on how big the potential case space is. Intuitively, given a fixed number of example cases, the bigger the potential case space, the smaller the IR measurement should be. In the system, we require that the IR measure for a relevancy rule must be greater than a certain threshold, which is termed as the E_level.

The following formula [4] is for the IR measurement,

$$IR_{cases}(rk) = \frac{M_c}{N_g \times 2^{s-N_c-n}} \qquad N_g \neq 0$$

$$IR_{cases}(rk) = 0 \qquad N_g = 0$$

where M_c is the number of examples satisfying the condition. The denominator is the potential number of all predictable cases satisfying the condition. N_g is the number of groups, if example cases satisfying the condition are divided according to the relevant set $P_1,..,P_n$. [5] The s is the number of independent predicates in the Description set, and N_c is the number of predicates in the condition. $s \geq N_c + n$.

3.4 Splitting The Relevancy Rule: Its Affects Upon The Measurements

Splitting relevancy rule has been discussed in section 2.3. If we have $rk = (Cond, \{P_1, P_2, ..P_n\}\ RR\ Q)$ Then

$$rk_{i_j} = (Cond_{i_j}, \{P_1, P_2, .., P_{i-1}, P_{i+1}, .., P_n\}\ RR\ Q)$$

are results of splitting rk by using P_i. Here the '$Cond_{i_j}$' is the conjunction of the old Condition '$Cond$' and a 'P_i op p_{i_j}' term, where p_{i_j} is a value of P_i, $j \in [1, n]$.

We will see that splitting an existing relevancy rule, because of its unsatisfactory T1 or IR measurements, is an important method of generating new relevancy rules with satisfactory measurements. In this subsection, properties will be presented concerning the relationship between the splitting of relevancy rules and its effects upon of the T1 and IR measurements.

From the definition of the T1 measurement, it can be shown [Huang, 1991c] that If we split rk into $rk_{i_1},..rk_{i_m}$, then

Property 4: $\quad Min_{j=1}^{m}\{T1(rk_{i_j})\} \leq T1(rk)$

rk_{i_j} is a new relevancy rule for $group_j$, which consists of all cases in which $P_i = p_{i_j}$.

This means that splitting the relevancy rule is a way of reducing the T1 measurement, or finding a new relevancy rule with a smaller T1 measurement.

From the definition of IR we can see that if a relevancy rule rk is broken into some more restricted ones, rk_i, thus partitioning the example cases into groups, the following unequation hold

Property 5: $\quad IR(rk) \leq Max\{IR(rk_i)\}$

This property says splitting a relevancy rule will at least increase the inductive reliabilities of some of the newly created relevancy rules.

Similarly, we can use P_k, which is not in the relevant set, to split a relevancy rule. The above two properties hold for this splitting as well.

3.5 Algorithm Outline

Given a set of example cases S, a description set D and a problem Q, this learning algorithm first sets the initial thresholds for ε and E_level, then it puts the root relevancy rule, ($Null$, D RR Q) into the result set R. The R is a set of candidate relevancy rules. The evaluation of the rule, the splitting of the rule and the process of reducing redundancy then follow.

The evaluation process decides whether a relevancy rule, rk, is acceptable or not. For every rule in R, it computes the three empirical measures, to see if

(i) $T1_S(rk) \leq \varepsilon$
(ii) $T2_S(rk) \leq 0$
(iii) $IR_S(rk) \geq E\_level$

If (i), (ii) and (iii) are all satisfied, this process keeps this rk in R. If either (iii) or (i) are not true, it performs a splitting process for this rk. If (i), (iii) are true, but not(ii), it attempts to reduce redundancy for this rk.

The splitting process takes a relevancy rule, rk = ($Cond$, P RR Q), split it into rk_i = ($Cond$ and $P_k = p_{k_i}$, P-{P_k} RR Q). P_k is chosen according to the general acceptance of all the rk_i, which is a function of the number of example cases and the potential new cases these acceptable rk_i cover.

Reducing redundancy for a rk = ($Cond$, P RR Q) is simply a question of removing the P_k, where P_k is the predicate corresponding to the maximum operation in the $T2$ measurement formula. If there are more than one of such P_i the one with the least variation in its values will be chosen. It will enable us to achieve the relevancy rule with the maximum generality.

3.6 Example

Based upon the strategies and evaluation measures, an experimental system of inductive learning of relevancy rules has been developed. Experiments has been carried out in the domains of a small block world, an animal taxonomy problem, and a national flag description problem. These results have shown that relevancy rules in most cases have a very clear and understandable meaning. Using these relevancy rules, the AR engine can solve new problems. More experiences have resulted in stronger relevancy rules, which will enable the AR engine to handle more new problems correctly. [Huang, 1991a;1991c]

In this section, this method is illustrated in a simple Block Moving Problem. Our problem in this block world will be briefly introduced. Then the relevancy knowledge learned from the experience is given. We will also see that using this knowledge, the problem can be solved correctly in certain cases.

This block world contains a three dimensional space, a block and a robot arm. Each dimension of the space has only two possible positions. The block may take one of two different shapes and may have one of two different

[4] This is a simplified formula. For details see [Huang, 1991c]

[5] $N_g = 1$ if $n = 0$ and $M_c \neq 0$. Actually $N_g = 0$ implies $M_c = 0$, and vice versa.

colours. The job of the robot arm is to move the block from an initial position to a target position.

There are two tools the robot arm may use, each corresponding to a shape of the block. The robot arm may move one position in one of six directions if possible. There are no explicit rules about which tool should be used or which action should be taken for a given problem case.

Our model is to learn the relevancy rules from the experiences supplied. These relevancy rules will enable the AR engine to guide the robot to solve a new problem analogically.

Table 4 describes one set of experiences

Initial Position					goal Position				Motor Directions		
X	Y	Z	C	S	X	Y	Z	T	U	L	F
P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0	0	1	1	1	n	r	n
0	0	1	0	0	0	1	1	1	n	n	f
0	0	0	0	0	0	1	0	1	n	n	f
0	0	0	0	1	0	1	1	0	n	r	n
0	0	1	0	1	0	1	1	0	n	n	f
0	1	0	0	0	0	0	0	1	n	n	b
0	1	0	0	0	0	1	1	1	n	r	n
0	0	1	1	0	0	1	1	1	n	n	f

Table 4 Training Experiences

where C = Colour; S = Shape; T = Tool; X (up-down), Y (forward-backward), Z (right-left) are coordinates; U = Up-Down Motor; L = Left-Right Motor; F = Forward-Backward Motor; r = right; l = left; f =forward; b = backward; u = up; d = down; n = null.

Applying the induction process, the following relevancies have been produced among others, with $\varepsilon = 0.1$, and E_level = 1/6. This E_level is quite small. This is because the examples we supply to the model represent a very small portion of all the possible cases.

rk_1, ($P_1 = 0$ and $P_6 = 0$ and $P_2 = 0$, $\{P_5\}$ RR Q_1)
rk_{1a}, ($P_1 = 0$ and $P_6 = 0$ and $P_8 = 1$, $\{P_5\}$ RR Q_1)
rk_2, ($P_1 = 0$ and $P_4 = 0$ and $P_6 = 0$, $Null$ RR Q_2)
rk_{2a}, ($P_1 = 0$ and $P_6 = 0$ and $P_7 = 1$, $Null$ RR Q_2)
rk_3, ($P_1 = 0$ and $P_6 = 0$, $\{P_3,P_8\}$ RR Q_3)
rk_4, ($P_1 = 0$ and $P_6 = 0$, $\{P_2,P_3,P_8\}$ RR Q_4)

These relevancy rules give some useful regularities within certain case sub-spaces. Rules rk_1 and rk_{1a} say that to decide which tool to use, only the shape of the block is relevant. Rules rk_2 and rk_{2a} say that Q_2 is a constant. This is because we do not have any experience of moving the block up and down. To decide the action of the left-right motor, rk_3 says that we only need to consider the Z coordinates of the initial and target cases. Rule rk_4 says that in order to make a forward-backward move, we need to consider not only the Y coordinate of the initial situation, but also the Z coordinates of the initial and the goal situations. This can be explained by the fact that the right-left motor has priority over the forward-backward motor.

Now let us use these relevancy rules for the testing cases in the Test Cases Table. Relevancy rule guided AR can be carried out in two directions: data directed or goal directed. The latter variant is described here.

Given a goal in a target case, the relevancy rule based AR process is as follow.

(i) Find a relevancy rule with the goal as its Q part and where the target case satisfies the condition part of the rule.

(ii) Find an experience case, which satisfies the 'condition', and shares the same values of properties in the relevant set of the relevancy rule with the target case .

(iii) Introduce the value of the goal in the experience case into the target. In complex situation, transformation is necessary. Building a tower analogically in a multiple blocks world is an example [Huang, 1991a].

(iv) Verify this value in the target case.

test	Init_Pos			C	S	Goal_Pos			Q_1	Q_2	Q_3	Q_4
a	0	0	0	0	1	0	1	0	0	n	n	f
b	0	0	0	1	0	0	1	0	1	n	n	f
c	0	0	0	1	1	0	1	0	0	n	n	f
d	0	0	0	1	1	0	1	1	0	n	r	n
e	0	0	0	1	0	0	1	1	1	n	r	n
f	0	0	1	1	1	0	1	1	0	n	n	f
g	0	1	0	0	0	0	1	0	1	n	n	n
h	0	0	1	1	1	0	1	0	0	n	?	f
j	0	0	1	0	0	0	1	0	1	n	?	f
k	0	1	0	0	1	0	0	0	0	?	n	b

Table 5 Test Cases Results

The results of applying this process and using the above relevancy knowledge are shown by table 5. Tests a to g have reached the correct conclusions. In Tests h and j, the model failed to generate actions for the Left-Right motor because of its lack of experience of moving left. In Test k it failed to generate the Up-Down motor action. The reason is that the model has not experienced enough cases in which the P_7 property takes 0 value and therefore has not generated a relevancy rule for Q_2 which applies to new cases with $P_7 = 0$.

4 Conclusion

Knowledge guided AR is a way of making AR justifiable. The knowledge used in AR is different from that of conventional deductive reasoning mechanisms. This paper proposes the relevancy rule as a form of knowledge which can support the AR process. Relevancy knowledge describes a kind of general relationship between the properties of a given world. This is in contrast with Greiner's abstract formula, which describes a strong theorem in an abstract domain applied to several different concrete domains. Automated acquisition of relevancy rules is easier than that of abstract formulae. Therefore the relevancy rule based AR model has a better chance of increasing its power. The relevancy rule considers the relevancy of a property for deciding another property is important. This is an enhancement of Russell's determination rule, which considers only determination information. Because considering irrelevant information may interfere with the analogical comparison. Relevancy rules enable an AR system to deal with new cases bet-

ter than a determination rule does. However, no attempt has been made to empirically or experimentally compare the results of relevancy rule based AR with that of any other methods, as that is beyond the scope of this paper.

Moreover, no comparison has been made between AR and conventional deductive methods. This is a field which lacks substantial work. So far, most authors consider AR to be a compliment to deductive methods. Only when the latter fails to produce results then AR comes into play. [Holland *et al.*, 1989] There is no justifiable reason for giving deductive methods priority over AR in solving problems. Research on the relevancy rule may shed light on the study of the relationship between AR and deductive methods. One clue is that the conventional form of production rule has only a trivial difference to a relevancy rule with a '*Null*' as it relevant set. Can we say that deductive reasoning is a special form of AR? Study of the evolution of relevancy knowledge may provide a hint for explaining why both schemes of reasoning exist and how they relate to each other. This explanation seems more profound than simply claiming AR and deductive methods are completely different and compliment each other.

Learning is an important field of study in AI. An intelligent system need to be able to learn, i.e. be capable of improving its ability automatically. The relevancy rule based AR system is no exception since successful reasoning in this system is based upon the availability of relevancy knowledge.

An important source of knowledge is experience. Inductive learning of relevancy rules is a method of acquiring relevancy laws from cases from experience. This paper has proposed a way of searching and evaluating candidate relevancy rules. The nature of this method is to select a promising candidate rule and check if it accounts for its experiences according to certain criteria.

In addition, This paper has defined three empirical measures for the relevancy rule. They are the sufficiency measure, the necessity measure and the inductive reliability measure. They first two correspond to the logical requirements of a relevancy rule. The third concerns the amount of evidence or experiences that support the candidate rule. The criteria are set, based upon these measures, for accepting a candidate relevancy rule.

The learning of relevancy rules is a new topic, conventional techniques, such as heuristics of non-exhaustive searching and the entropy measurement method, are used in the experimental inductive system. However, some unanswered questions remain. How should the thresholds for accepting a relevancy rule be decided in the first place? How should they be adjusted according to the performance of the system in dealing with new cases? At present, initial thresholds are set according to the proportion of exceptional cases to the whole case (for the ε) and according to a prediction rate (for the *E_level*). For instance, if we allow one exception in every ten cases on average, the ε would be 0.9log(0.9)+0.1log(0.1). Adjustment of the thresholds can be based upon the error rate of the analogical arguments. New thresholds should result in the evolution of relevancy rules. Substantial work is needed in this area.

To conclude, this paper has shown that relevancy knowledge is a form of knowledge which supports justifiable AR. The inductive learning method described here provides a way of automatic acquisition of relevancy knowledge. This increases the ability of the relevancy rule guided AR system. The work described in this paper provides a preliminary basis for further study.

Acknowledgements

The first author wishes to thank all parties of the Sino-British Friendship Scholarship Scheme for sponsoring his studies resulting this paper. He would also like to thank Gerard V. Conroy for helpful discussions.

References

[Greiner, 1988a] Russell Greiner. Abstraction-Based Analogical Inference. In David H. Helman (Ed.) *Analogical Reasoning*, Kluwer Academic Publisher, 1988.

[Greiner, 1988b] Russell Greiner. Learning by Understanding Analogies . In Armand Priedits(Ed.), *Analogica*, Pitman Publishing, London, 1988.

[Holland, *et al.* , 1989] John H. Holland, Keith J. Holyoak, Richard E. Nisbett and Paul R. Thagard. *Induction : Processes of Inference, Learning, and Discovery.* MIT, London, 1989.

[Huang, 1991a] Ye Huang. Analogical Reasoning: The Use of Relevancy Rule. *Internal Report*, Computation Department, UMIST, UK, 1991.

[Huang, 1991b] Ye Huang. On Derivation of Relevancy Knowledge. *Internal Report*, Computation Department, UMIST, UK, 1991.

[Huang, 1991c] Ye Huang. On Inductive Acquisition of Relevancy Knowledge. *Internal Report*, Computation Department, UMIST, UK, 1991.

[Kedar-Cabelli, 1988a] Smadar Kedar-Cabelli. Analogy – From a Unified Perspective. In David H. Helman (Ed.), *Analogical Reasoning*, Kluwer Academic Publisher, 1988.

[Kedar-Cabelli, 1988b] Smadar Kedar-Cabelli. Toward a Computational Model of Purpose-Directed Analogy. In Armand Priedits (Ed.), *Analogica*, Pitman Publishing, London, 1988.

[Quinlan, 1983] Ross Quinlan. Learning Efficient Classification Procedures. In R.S. Michalski, J.G. Carbonel and T.M. Mitchel (Eds.), *Machine Learning: An Artificial Intelligence Approach, Ch. 15, Vol I*, Tioga Press, Palo Alto, CA, 1983.

[Russell, 1988] Stuart J, Russell. *The Use of Knowledge in Analogy and Induction.* Pitman Publishing, London, 1988.

[Vosniadou and Ortony, 1989] Stella Vosniadou and Andrew Ortony. *Similarity and Analogy.* Cambridge Press, 1989.