CSCSI-90

PROCEEDINGS OF THE EIGHTH BIENNIAL CONFERENCE OF THE CANADIAN SOCIETY FOR COMPUTATIONAL STUDIES OF INTELLIGENCE

ACTES DE LA HUITIÈME CONFÉRENCE BIENNALE DE LA SOCIÉTÉ CANADIENNE POUR L'ÉTUDE DE L'INTELLIGENCE PAR ORDINATEUR

edited by/sous la direction de Peter F. Patel-Schneider

University of Ottawa
Ottawa, Ontario, Canada
22–25 May 1990

Université d'Ottawa
Ottawa, Ontario, Canada
22–25 mai 1990

PROCEEDINGS OF THE EIGHTH BIENNIAL CONFERENCE OF THE CANADIAN SOCIETY FOR COMPUTATIONAL STUDIES OF INTELLIGENCE

ACTES DE LA HUITIÈME CONFÉRENCE BIENNALE DE LA SOCIÉTÉ CANADIENNE POUR L'ÉTUDE DE L'INTELLIGENCE PAR ORDINATEUR

edited by/sous la direction de Peter F. Patel-Schneider

University of Ottawa
Ottawa, Ontario, Canada
22–25 May 1990

Université d'Ottawa
Ottawa, Ontario, Canada
22–25 mai 1990

Sponsored by/Parrainée par:
Canadian Society for Computational Studies of Intelligence/
Société canadienne pour l'étude de l'intelligence par ordinateur
National Research Council of Canada/
Conseil national de recherches du Canada

Supported by/Avec l'appui financier de:
National Sciences and Engineering Council of Canada/
Conseil de recherches en sciences naturelles et en génie du Canada
PRECARN Associates/Associés PRECARN
Bell-Northern Research/Recherches Bell-Northern
Government of Ontario/Gouvernement de l'Ontario

In cooperation with/Et la collaboration de:
University of Ottawa/Université d'Ottawa

ISBN 0-9694596-0-2

©1990
Canadian Society for Computational Studies of Intelligence
Société canadienne pour l'étude de l'intelligence par ordinateur

Edited by/Sous la direction de Peter F. Patel-Schneider

Printed by/Imprimé par IMPRICO, Ville Mont-Royal, CANADA

Copies of these proceedings may be obtained as follows:

In Canada:
$35 CDN each (CSCSI member rate)
$40 CDN each (non-member rate)
Add $5 CDN for postage
Send orders together with payments to:
CIPS
243 College Street (5th floor)
Toronto, Ontario M5T 2Y1
CANADA

Outside Canada:
$35 US each (California residents add sales tax.)
In the United States, add $2 US for the first copy and $0.75 US for each additional copy for shipment by book rate. Outside the United States, add $3 US for the first copy and $2 US for each additional copy for shipment by surface mail and include payment with order.

Send orders to:
Morgan-Kaufmann Publishers, Inc.
Order Fulfillment Center
P. O. Box 50490
Palo Alto, California 94303
U.S.A.

On peut obtenir des exemplaires du présent actes en procédant comme suit:

Au Canada:
35 $ can. chacun (membres de la SCEIO)
40 $ can. chacun (non-membres)
Ajouter 5 $ can. pour frais de poste
Adresser les demandes accompagnées du paiement à:
ACI
243, rue College (5^e étage)
Toronto, Ontario M5T 2Y1
CANADA

En dehors du Canada:
35 $ US chaqun (Les résidents de la Californie doivent ajouter la taxe de vente d'Etat.)
Aux États-Unis, ajouter 2 $ US pour le premier exemplaire et 0,75 $ US pour chaque exempliare additionel expédié par post-tarif livre. En dehors des États-Unit, veuillez ajouter 3 $ US pour le premier exemplaire et 2 $ US pour chaque exemplaire supplémentaire expédié par courrier ordinaire. Le tout est payable à l'avance.

Adresser les demandes à:
Morgan-Kaufmann Publishers, Inc.
Order Fulfillment Center
P. O. Box 50490
Palo Alto, California 94303
U.S.A.

Message from the General Chairman
Message du président de la conférence

Dick Peacocke, Bell-Northern Research/Recherches Bell-Northern

Welcome to the Eighth Canadian Conference on Artificial Intelligence, which is being held on 22–25 May 1990 at the University of Ottawa. This is the first time the conference has been held in Ottawa, although the first CSCSI/SCEIO workshop did take place here in 1975.

The conference programme will consist of both invited and contributed papers presented in a plenary session format over three full days. A complete set of the papers appears in these proceedings.

A new feature at this conference is the inclusion of four tutorial sessions which will be held on Tuesday, 22 May, 1990 on the following topics:

- Intelligent Computer Aided Software Engineering Tools
- Machine Learning
- Expert Database Systems
- Robotics from an AI Perspective: Planning and Control

These tutorials are organized with the view of providing an overview of the state of technological developments, examining current applications, and discussing future needs and directions.

I wish to express my warmest thanks to the members of the Organizing Committee, to the Chairman and members of the Program Committee, to Conference Services (NRC), and last but not least, to Ken Charbonneau for his attention to all the details.

I would like to extend my invitation to you for the conference. Ottawa is particularly beautiful in the month of May. Over a million tulips are blooming by the Rideau Canal. Enjoy the architecture, parks, restaurants, and entertainment in the National Capital Region, as well as the conference.

Je vous souhaite la bienvenue à la Huitième Conférence Canadienne sur l'Intelligence Artificielle qui aura lieu du 22 au 25 mars 1990 à l'Université d'Ottawa. C'est la première fois que cette conférence se déroule à Ottawa bien que le premier atelier de CSCSI/SCEIO ait eu lieu ici en 1975.

Le programme comprendra des communications à titre de conférenciers invités et à titre de propositions acceptées. Ces communications seront présentées dans le cadre de séances plénières sur une période de trois jours pleins. Toutes les communications présentées sont publiées dans ce volume.

Une nouvelle caractéristique de cette conférence est l'introduction de quatre cours d'instruction qui se tiendront le mardi 22 mai 1990 sur les sujets suivants:

- Outils intelligents de génie logiciel assisté par ordinateur
- Apprentissage automatique
- Systèmes experts et bases de données
- La robotique du point de vue de l'IA: Planification et contrôle.

Ces cours sont offerts dans le but de fournir un aperçu de l'état de la technologie et des développements récents, d'en analyser les applications courantes et de discuter des besoins et directions à venir.

Je tiens à exprimer mes remerciements les plus chaleureux aux membres du comité organisateur ainsi qu'aux services de conférence (CNR) mais surtout à Ken Charbonneau pour son soucis de tous les détails.

Je vous invite à participer à la conférence et à profiter de l'occasion pour découvrir Ottawa qui, en mai, est une ville particulièrement belle: plus d'un million de tulipes fleurissent le long du canal Rideau. Venez aussi découvrir l'architecture, les parcs et les restaurants de la région de la Capitale Nationale.

CSCSI-90 Organizing Committee
Comité organisateur de la SCEIO-90

Erratum:

Robert Mercer and Dick Peacocke were inadvertently left off the listing of the program committee for the Seventh Biennial Conference of the Canadian Society for Computational Studies of Intelligence.

Par inadvertance, les noms de Robert Mercer et de Dick Peacocke manquent sur la liste des membres du comité du programme de la Septième conférence de la Société canadienne pour l'étude de l'intelligence par ordinateur.

Message from the Program Chairman
Message du président du comité du programme

Peter F. Patel-Schneider, AT&T Bell Laboratories

This year 112 papers were submitted to the conference, and received in time to be reviewed. Seventeen papers were received too late to be reviewed. It is unfortunate that such a large number of papers had to be turned away, but the deadline for receipt of papers was made as late as possible to allow for current results to be presented at the conference. This meant that the review period was short, and that late papers could not be considered for review. Most papers were from Canada and the U. S. A., but papers were also submitted from Australia, Bulgaria, England, the Federal Republic of Germany, France, India, Israel, Korea, the Netherlands, Northern Ireland, Japan, the People's Republic of China, Portugal, Scotland, and Sweden giving the conference a decidedly international flavour. Of the 112 papers reviewed, 37 papers were accepted, and appear in this volume.

The large number of submissions received this year meant that many submissions were not accepted. This is unfortunate, as it means that some interesting papers will not be presented at this conference. One fortunate effect of the selectivity exhibited by the program committee is that the conference can remain in a single-track format. This format allows attendees to listen to all the talks at the conference, instead of concentrating only on talks in their speciality, and not hearing about recent developments in other areas of AI.

Cette année 112 communications furent envoyées au comité du programme, et reçues assez tôt pour être soumises à l'arbitrage. Dix-sept communications furent reçues trop tard pour être evaluées. Il est dommage qu'autant de communications dussent être renvoyées, mais le délai de réception des communications fut fixé aussi tard que possible, pour permettre la présentation de résultats actuels lors de la conférence. La période de rédaction des évaluations fut donc brève, et il fut impossible de tenir compte des communications en retard. La plupart des communications provinrent du Canada et des États-Unis, mais des communications furent aussi reçues en provenance d'Angleterre, d'Australie, de Bulgarie, de Corée, d'Écosse, de France, d'Inde, d'Irlande du Nord, d'Israel, du Japon, des Pays-Bas, du Portugal, de la République Fédérale d'Allemagne, de la République Populaire de Chine, et de Suède, donnant ainsi à la conférence une allure décidément internationale. Des 112 communications évaluées, 37 communications furent acceptées, et sont publiées dans ce volume.

Le grand nombre de soumissions reçues cette année veut dire que beaucoup ne furent pas acceptées. Cette circonstance est malheureuse, car elle signifie que certaines communications intéressantes ne seront pas données lors de la présente conférence. Un effet bienvenu du degré de sélection exercé par le comité du programme est que la conférence peut garder un format à piste unique. Ce format permet à tous les participants d'assister à tous les exposés donnés à la conférence, au lieu de se concentrer sur les exposés dans leur spécialité, sans entendre parler des développements récents dans d'autres branches de l'IA.

CSCSI Executive Committee 1988–1990
Comité exécutif de la SCEIO 1988–1990

CSCSI-90 Program Committee
Comité du programme de la SCEIO-90

Program Chairman/Président du comité

Peter F. Patel-Schneider, AT&T Bell Laboratories

Program Committee/Comité du programme

Fahiem Bacchus, Department of Computer Science, University of Waterloo
Nicholas J. Cercone, Centre for Systems Science, Simon Fraser University
Robin Cohen, Department of Computer Science, University of Waterloo
Renato De Mori, School of Computer Science, McGill University
Renee Elio, Department of Computer Science, University of Alberta
Charles Elkan, Department of Computer Science, University of Toronto
Brian V. Funt, School of Computing Science, Simon Fraser University
William Havens, School of Computing Science, Simon Fraser University
Jim Greer, Department of Computational Science, University of Saskatchewan, Saskatoon
Russell Greiner, Department of Computer Science, University of Toronto
Robert E. Mercer, Department of Computer Science, University of Western Ontario
Jan A. Mulder, Department of Mathematics, Statistics, and Computing Science, Dalhousie University
Eric Neufeld, Department of Computational Science, University of Saskatchewan, Saskatoon
David Poole, Department of Computer Science, University of British Columbia
Alberto Segre, Department of Computer Science, Cornell University
Edward P. Stabler, Jr., Department of Linguistics, University of California, Los Angeles

Referees/Arbitres

Bill Armstrong
Robert Bergevin
Mike Dawes
James P. Delgrande
Marc Dymetman
David W. Etherington
Daniel Fass
Randy Goebel
Scott Goodwin
Chris Groeneboer
Robert F. Hadley
Gary Hall
V. Hayward
Diane Horton
Xueming Huang
Pierre Isabelle
Mark A. Jones
Anthony Kusalik
Ze-Nian Li
Gordon I. McCalla
Carl McCrosky
Paul McFetridge
Evangelos Milios
Jeff Pelletier
Fred Popowich
Robert Prager
Gerhard Roth
Dale Schuurmans
Bart Selman
Lingyan Shu
Fei Song
Bruce Spencer
Devika Subramanian
Jennifer S. Turney
Peter van Beek
Carl Vogel
Raymond L. Watrous
Patrick G. Xavier
Qiang Yang
Jia-Huai You

Invited Speakers
Conférenciers invités

Johann de Kleer, System Sciences Laboratory, Xerox Palo Alto Research Center

Circumscribing the Artificial Engineer: Diagnosis

Brian Funt, School of Computing Science, Simon Fraser University

Colours from Colour Signals

Graeme Hirst, Department of Computer Science, University of Toronto

Planning the Future of Natural Language Research (even in Canada)

Tom M. Mitchell, Department of Computer Science and Robotics,
Carnegie Mellon University

Can We Build Learning Robots?

Best Paper Award
Prix de la meilleure communication

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 Rina Dechter, for "From Local to Global Consistency", and André Trudel, for "Temporal Integration".

Le prix SCEIO 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 du programme de la conférence SCEIO 1990 est heureux de décerner le prix de la meilleure communication à Rina Dechter et à André Trudel.

Table of Contents
Table des matières

WHICH IS MORE BELIEVABLE, THE PROBABLY PROVABLE OR THE PROVABLY PROBABLE?*

Judea Pearl
Cognitive Systems Laboratory
UCLA Computer Science Department
Los Angeles, CA 90024-1596

Abstract

This paper describes and compares two paradigms for processing incomplete specifications of probabilistic knowledge. The first computes provable probability statements by treating the specifications as constraints over probabilities. The second computes how probable it is that a proposition is provable, treating the specifications as randomly sampled assumptions added onto logical theories. We first examine the representational power of the sampled-assumptions paradigm and then we identify and assess two of its major shortcomings: Failing to represent dependencies among events with unknown probabilities, and failing to represent domain knowledge cast in the form of defeasible conditional sentences.

1. Introduction

Consider the following problem: We are given a set F of propositional formulas, each formula $f \in F$ is assigned a degree of certainty $P(f)$, and we are asked to determine how strongly one should believe in some other formula q (q stands for "query").

We purposely phrased the problem using the vague terms "certainty" and "belief," since problems of this nature permit a variety of interpretations. Had the set of specified certainties $S = \{P(f): f \in F\}$ been sufficient for defining a coherent probability function P on the models of the language (by a model we mean a truth valuation of all literals) the answer would then be given simply by equating "belief in q" with $P(q)$. But if the input information is insufficient, two approaches are feasible, representing two complementary conceptions of partially specified knowledge. In one, we consider S as a set of properties that a probability function should satisfy or, equivalently, as a set of constraints over an implicit family $\mathbf{P}$ of coherent probability functions. The answer to our problem would then be given in a form of an interval

$$P_*(q) \le P(q) \le P^*(q)$$

where P_* and P^* represent the lowest and highest value that $P(q)$ can attain by any member of $\mathbf{P}$. The second alternative is to regard S as a policy for selecting assumptions (or axioms) from F and examining their logical consequences. Given a probability distribution over a set F of assumptions, our problem can be interpreted as that of assessing the certainty that q is provably true, namely, the probability that an assumption (or a set of assumptions) be selected, from which a proof of q can be assembled.

We will denote the first interpretation by $P_*(q)$ and the second by $Bel(q)$. Formally,

$$P_*(q) = \min\{P(q): P \in \mathbf{P}\} = \max\{t: P(q) \ge t\}$$

and

$$Bel(q) = P(f: f \supset q)$$

Thus, P_* measures the highest level that we can "provably" attribute to $P(q)$, while Bel measures the probability that q is provable. To contrast this difference syntactically, we can make an unorthodox usage of the symbol $\models$ to denote "it is provably true that ...", and write:

* This work was supported in part by National Science Foundation Grant #IRI-8821444 and the Naval Research Laboratory Grant #N00014-89-J-2007.

$$P_*(q) = \max \{ t : \models [P(q) \geq t] \}$$

$$Bel(q) = P(\models q)$$

$P_*(q)$ uses probability theory as the object language and logic as a meta-language; $Bel(q)$ reverses these roles. (1)

Historically, the lower probability measure P_* has been studied by Bayesians philosophers such as de Finetti [1974], Good [1950], and Smith [1961] and more recently introduced into the AI literature by Nilsson [1986]. The function Bel, on the other hand, corresponds to the measure developed by Dempster [1967] and Shafer [1976] under the name "belief functions", and has recently been given an ATMS formulation [de Kleer 1986] by attaching probabilities to the ATMS assumptions [Laskey and Lehner 1989; Provan 1989].

The purpose of this paper is two-fold:

1. To characterize the notions of probably-provable vs provably-probable, illustrate their semantic differences and highlight their distinctive patterns of behavior.

2. To compare the expressional power of these two conceptions and assess their adequacy as representations of incomplete knowledge and uncertain evidence.

2. An Illustration

A natural question to ask is whether it makes a substantial difference how we formulate a problem; in terms of assumptions about probabilities or probabilities about choosing assumptions. A second question is whether every problem of incomplete knowledge can be conveniently represented in either one of the two formulations. Example 1 illustrates these two issues.

Example 1: The Peter, Paul and Mary Sandwich

Mary challenges Peter to guess what kind of sandwich she happened to prepare for lunch that day, ham or turkey. She also promises to pay Paul 1,000 dollars if Peter guesses correctly. Peter says that, for lack of even the slightest clue, he is going to toss a fair coin and guess "ham" if it turns up heads, "turkey" if it turns up tails. Mary asks Paul if he is not anxious to know what sandwich she actually prepared, but Paul brushes her off saying that he already had lunch and that it makes no difference to him; regardless of whether it is ham or turkey, in either case he has exactly a 50% chance of winning the 1,000 dollars.

Mary retorts that Paul is behaving like an incurable Bayesian, and that instead of considering the chances of winning, he should be considering the chances that winning is ASSURED by the specific evidence at hand, namely, by Peter's guessing policy. She claims that Paul's current "belief" of winning is, in fact, zero, because either outcome of the coin, heads or tails, would leave him with no assurance of winning. However, if he would only listen to her for a moment, his belief would immediately jump to 1/2, because, knowing what kind of sandwich it is would give him a 50% assurance of winning.

Paul answers that he gets enough assurance just thinking about Mary's sandwich: "If I have a 50% assurance assuming it is ham, and 50% assuming it is turkey, then I have a 50% assurance, period!"

Mary does not give up: "No, Mr. Wise Guy, you can't have a 50% assurance of winning, because it leads to a paradoxical conclusion: If you win, you can do it in one of two ways, either matching heads with ham or tails with turkey, with equal chance to each way. Similarly if you lose, you either mismatched heads with turkey or tails with ham, with equal chances. Thus, having a 50% assurance of winning permits you to conclude that there is a 50% chance that the sandwich I made is ham while, in fact, you know nothing about my sandwich."

The sandwich story illustrates two points. First, it *does* make a qualitative difference how we interpret "degree of belief," as a provable probability (P_*) or as the probability of provability (Bel), with each interpretation leading to a different action and a different information gathering strategy. The P_* interpretation proclaims Mary's information (regarding the sandwich) useless, while the Bel interpretation values it as useful, capable of lifting one's belief from zero to ½ regardless of the outcome.

This feature is characteristic to the probably-provable interpretation of beliefs, because it is quite possible that a proposition A is not provable from any one of the sampled assumptions, thus rendering $Bel(A) = 0$, but if we add either B or $\neg B$ as an axiom, then A will be provable under some assumption (through not the same), thus rendering both $Bel(A \mid B)$ and $Bel(A \mid \neg B)$ greater than zero. Whereas the value of $P_*(A)$ must be "sandwiched" somewhere between $P_*(A \mid B)$ and $P_*(A \mid \neg B)$, $Bel(A)$ might violate this principle(2) and satisfy

$$Bel(A) < \min [Bel(A \mid B),\ Bel(A \mid \neg B)] .$$

Consequently, decision strategies based on the magnitude of $Bel(\cdot)$ might exhibit peculiar behavior, such as the chasing

(1) J. Halpern (in conversation) has pointed out that the logics used in the two paradigms are not the same; the former uses the axioms of probabiity theory to deduce assertions about probability inequalities, while the latter uses propositional logic as the object language. Likewise, the probabilities in the two paradigms are not defined on the same space; in the former, probabilities are defined on propositions, while in the latter, probabilities are defined over logical theories. Note also that our notion of "provability" is semantical (being a logical consequence) rather than syntactical and is independent, therefore, of the axiomatic system used.

(2) The "sandwich" metaphor is due to Aleliunas [1988] and was termed the "Principle of the hypothetical middle" in Pearl [1988].

after useless information sources.

The second feature demonstrated by the sandwich story is that the task of encoding partial knowledge in terms of randomly chosen assumptions is not as easy as it might seem. The assignment of probabilities to some propositions often induces definite probabilities on other propositions and one may be faced with an unresolvable dilemma of whether to add those other propositions to the set of assumptions or not. If we leave them out, the analysis might never recover the information lost. If we let them in, we must decide how to combine them with other assumptions, and any such decision might produce spurious conclusions, pretending to knowledge we do not in fact have.

In our example, since Peter's coin is independent of Mary's sandwich, the assertion $P(win) = ½$ follows as a straight forward consequence of $P(heads) = ½$. The question is how to encode these two items of information as a procedure for sampling assumptions.(3) If we encode only one item, say $\{P(heads) = ½, P(tail) = ½\}$, the other will not be recovered correctly; $Bel(win)$ computes to zero instead of ½, because there is no way to prove "win" from either "heads" or "tail". If we try to encode both items, we do not know with what probability the joint assumption $heads \wedge win$ should be sampled and, whatever we assume for this joint probability, we find that we suddenly know more than we should about the third item, the sandwich. For example, assuming (as Mary did in the last paragraph of Example 1) that "win" and "heads" are to be chosen independently of each other, the probability of proving "ham" calculates to ½ while in reality we have no information about Mary's sandwich.

A mathematical basis for recognizing when partial knowledge is encodable as random assumptions has been developed in the literature on belief functions (though not from this perspective nor with this terminology) and will be summarized next.

3. Mathematical Summary

Belief functions result from assigning probabilities to sets rather than to the individual points, with points representing specific worlds and sets reflecting propositions about those worlds. Given an initial probability assignment $m(\cdot)$ to a select set F of propositions (called *focal elements*), namely,

$$\sum_{B \in F} m(B) = 1 \,, \quad m(B) \geq 0 \,, \tag{1}$$

every proposition in the language then acquires a pair of measures, $Bel(\cdot)$ and $Pl(\cdot)$, such that

$$Bel(A) = \sum_{B \supset A} m(B) \tag{2}$$

and

$$Pl(A) = 1 - Bel(\neg A) \,.$$

Any measure $Bel(\cdot)$ constructed in such a manner is called a *belief function*, and its associated measure $Pl(\cdot)$ is called *plausibility*.

A necessary and sufficient condition for a function $Bel(\cdot)$ to be a belief function is that it satisfies:

$$Bel(\varnothing) = 0, \quad Bel(A \vee \neg A) = 1 \text{ , and}$$

$$Bel(A_1 \vee \ldots \vee A_n) \geq \sum_i Bel(A_i) - \sum_{i<j} Bel(A_i \wedge A_j) + - \ldots \tag{3}$$

$A_1, A_2, \ldots, A_n$, being any collection of propositions.

Given two belief functions Bel_1 and Bel_2, their orthogonal sum $Bel_1 \oplus Bel_2$, also known as Dempster's rule of combination, is defined by their associated probability assignments

$$(m_1 \oplus m_2)(A) = K \sum_{A_1 \wedge A_2 = A} m_1(A_1) m_2(A_2) \quad A \neq \varnothing \tag{4}$$

where

$$K^{-1} = \sum_{A_1 \wedge A_2 \neq \varnothing} m_1(A_1) m_2(A_2) \tag{5}$$

The operator $\oplus$ is known to be commutative and associative.

As a special case of Eq. (4), if m_2 establishes the truth of proposition B, i.e., $m_2(B) = 1$, the combined belief functions becomes

$$Bel_1(A \mid B) = \frac{Bel_1(A \vee \neg B) - Bel_1(\neg B)}{1 - Bel_1(\neg B)} \tag{6}$$

This formula is known as Dempster's conditioning.

A belief function is called *additive* or *Bayesian* if each of its focal elements is a singleton, i.e., an elementary event or a possible world. Bayesian belief functions satisfy $Bel(A) = Pl(A) = 1 - Bel(\neg A)$. If Bel_1 is Bayesian, then $Bel_1 \oplus Bel_2$ is also Bayesian, and Dempster's conditioning reduces to ordinary Bayesian conditioning [Shafer 1976].

(3) The third variable, Mary's sandwich, does not qualify as an assumption because it is not given a definite probability.

4. Belief Functions and the Sampled-Assumptions Paradigm

The correspondence between belief functions and the sampled-assumptions paradigm is made clear in Eqs. (1) and (2). We are given a collection of logical theories, $T_1, \ldots, T_n$; each theory is characterized by an assumption formula $B \in F$ corresponding to one focal element and each theory is assigned a probability $P_i = m(B)$, such that the sum of the probabilities is 1. The belief in a formula A is the sum of the probabilities of the theories from which A follows as a logical consequence. Note that the basic probability assignment $m(B)$ in Eq. (1) does not specify the net overall probability of B, since the truth of B may be implied by other focal elements as well. Instead, it specifies the probability that the theory defined by B *alone* is adopted, and accordingly, $Bel(A)$ represents the probability that A is provable in some randomly adopted theory.

This interpretation provides a simple semantics for Dempster's rule, Eq. (4), which has been used extensively for combining independent pieces of evidence. Each piece of evidence, say e_1 and e_2, defines a probability mass over a collection of potentially adaptable theories, F_1 and F_2, and the combined evidence $e_1 \oplus e_2$, likewise, defines a probability mass over a collection of joint theories. Each joint theory is characterized by the conjunction of two assumptions, one sampled from F_1 and one from F_2. The mass assigned to such conjunction is the product of the individual masses (thus reflecting evidence independence), while the mass attributed to any contradictory theory is redistributed among the non-contradictory theories in proportion to their weights. Thus, the belief function resulting from this combination rule is simply the *conditional probability of provability*, given that the two pieces of evidence are noncontradictory [Pearl 1988].

5. Encoding Probabilistic Specifications as Sampled Assumptions

A *specification* is any assertion (or constraint) about properties of a probability function, for example, $P(A \wedge B) = p, P(B \mid A) = q \;, P(A \mid B \wedge C) = P(A \mid B)$, $P(B) > P(A)$, etc. Let S be a set of specifications and let $\mathbf{P}_S$ be the set of all (additive) probability functions that satisfy S. A family $\mathbf{P}$ of probability distributions is said to be *compatible* with a belief function Bel, if for every proposition A, we have

$$Bel(A) = \min \{P(A) \colon P \in \mathbf{P}\} \triangleq P_*(A)$$

A set of specifications S is said to be *SA-encodable* ("SA" standing for "sampled-assumptions") if there exists a belief function that is compatible with $\mathbf{P}_S$.

It is well known [Dempster 1967] that, while every belief function has a compatible family of probability functions, the converse is not true; there are families of probability functions that have no compatible belief function. This means that certain types of probabilistic specifications, corresponding to certain types of partial knowledge, cannot be expressed in the language of randomly chosen assumptions. Examples of such cases presenting common types of partial knowledge will be given next.

5.1 The Representation of Unknown Interactions

Example 2: We have two events, E_1 and E_2. We know their individual probabilities, $P(E_1) = P(E_2) = ½$, but we know nothing about their interaction. The specification set in this case is

$$S = \{ P(E_1) = ½,\; P(E_2) = ½\},$$

which permits the probability of each of the four joint events $\{E_1 \wedge E_2, E_1 \wedge \neg E_2, \neg E_1 \wedge E_2, \neg E_1 \wedge \neg E_2\}$ to range between 0 and ½. Thus, $P_* = 0$ and $P^* = ½$ for each of these joint events. This specification set is not SA-encodable because any assignment of zero belief to four individual points and, simultaneously, a belief of ½ to four pairs of these points (as required by S) would violate Eq. (3).

The failure to represent ignorance about interactions does not present a severe limitation on the practical applications of the SA paradigm. Faced with such ignorance the naive user would normally encode S as two separate belief functions:

$$m_1(E_1) = ½ \qquad m_2(E_2) = ½$$

$$m_1(\neg E_1) = ½ \qquad m_2(\neg E_2) = -½$$

which, combined by Dempster's rule, would yield a probability mass of *quarter* for each of the four joint events. This amounts to assuming that the assumptions E_1 and E_2 are sampled independently of each other (to form joint theories) as if E_1 and E_2 were known apriori to be independent events. The fact that independence was not really part of the specification set is not too disturbing because it conforms to the discourse convention that, unless warned otherwise, events can be presumed to be independent of each other. This convention underlies most work in default reasoning and can also be traced to the maximum-entropy principle [Pearl 1989].

5.2 The Representation of Independent Events

Example 3: We have two independent events, E_1 and E_2. We know the unconditional probability $P(E_1) = ½$, but we know nothing about E_2, except its being independent of E_1.

The specification set is

$$S = \{P(E_1) = \tfrac{1}{2},\ P(E_1 | E_2) = P(E_1)\},$$

which corresponds exactly to the knowledge available in the sandwich story of Example 1 (with $E_1 = heads$ and $E_2 = ham$). S permits $P(E_2)$ to range over the interval [0, 1], and also dictates definite probabilities on certain formulae involving E_2, for example,

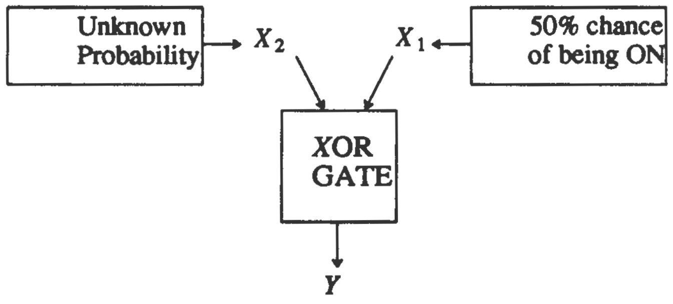
$$P[(E_1 \wedge E_2) \vee (\neg E_1 \wedge \neg E_2)] = \tfrac{1}{2}$$

and

$$P[(E_1 \wedge \neg E_2) \vee (\neg E_1 \wedge E_2)] = \tfrac{1}{2}.$$

No belief function exists which is compatible with these equalities, while simultaneously reflecting our state of ignorance about E_2, namely, $Bel(E_2) = 0$. Thus, S is not SA-encodable.

The limitation shown in Example 3 represents a more serious impediment to the applications of the sampled-assumptions approach. The sandwich story of Example 1 shows indeed that failing to represent $Bel[(E_1 \wedge E_2) \vee (\neg E_1 \wedge \neg E_2)] = Bel(win) = \tfrac{1}{2}$ can lead to a major clash with intuition. This transcends to practical problems as well. Consider a circuit diagnosis system using the SA-TMS approach, in the spirit of Laskey and Lehner [1989] or Provan [1989]. Imagine that we need to calculate the belief that the output Y of an exclusive-OR gate is ON, knowing that one of the inputs has a 50% chance of being ON, $P(X_1 = \text{ON}) = \tfrac{1}{2}$, while the other input, X_2, is totally unknown (see Figure 1).



Find: Belief (Y is ON)

Figure 1

The TMS engineer will now face the dilemma we discussed in the sandwich story: What propositions should be considered as assumptions? The naive approach would be to take as assumptions only propositions that are assigned explicit probabilities, namely, $X_1 = \text{ON}$ and $X_1 = \text{OFF}$. Sampling these two assumptions with 50% probability each yields:

$$Bel(Y = \text{ON}) = 0 \qquad Bel(Y = \text{OFF}) = 0$$
$$Bel(X_2 = \text{ON}) = 0 \qquad Bel(X_2 = \text{OFF}) = 0$$

This result ignores the information that X_1 and X_2 are independent, which should yield $P(Y = \text{ON}) = \tfrac{1}{2}$ regardless of the value of $P(X_2 = \text{ON})$.

In case the TMS engineer becomes aware of the inevitability of $P(Y = \text{ON}) = \tfrac{1}{2}$, and wishes to include it in the set of sampled assumptions, the SA-TMS will produce a paradoxical result regarding X_2; sampling $X_1 \in \{\text{ON}, \text{OFF}\}$ and $Y \in \{\text{ON}, \text{OFF}\}$ independently (giving 50% chance to each choice) yields $Bel(X_2 = \text{ON}) = \tfrac{1}{2}$. Moreover, regardless of the probability value by which we choose to sample the joint assumption $(X_1 = \text{ON}) \wedge (Y = \text{ON})$, we always get the equality $Bel(X_2 = \text{ON}) = 1 - Bel(X = \text{OFF}) = \text{Pl}(X_2 = \text{ON})$. This corresponds to having precise knowledge of $P(X_2 = \text{ON})$, which contradicts our starting hypothesis that $P(X_2 = \text{ON})$ is totally unknown. In large circuits, where X_2 may serve as an input and an output of other components as well, this might lead to erroneous predictions and diagnoses. For example, if $X_2 = \text{ON}$ signifies the failure of a component (of which X_2 is the output), the calculation of $Bel(X_2 = \text{ON}) = \tfrac{1}{2}$ may trigger an action to replace that component while, in reality, we possess no evidence whatsoever to that effect, since X_1 and X_2 were presumed independent.

5.3 The Representation of Conditional Information

Example 4: We are given a specification of two conditional probabilities,

$$S = \{P(A | B) = p, P(A | \neg B) = q\}, \qquad (7)$$

$$0 < 1 - q \le p \le q < 1,$$

but we are not given any of the unconditional probabilities. The information given in (7) induces several constraints on the probabilities of other propositions, including for example:

$$0 \le P(A \wedge B) \le p$$

$$p \le P(A) \le q$$

$$1 - q \le P[(A \wedge B) \vee (\neg A \wedge \neg B)] \le p$$

Again, no belief function exists that matches these upper and lower probabilities without violating the basic conditions in (3). Thus, S is not SA-encodable.

Example 4 reveals a second limitation of the random-assumptions model, showing it incapable of representing the specification of conditional probabilities. This means that large fragments of empirical knowledge cast in the form of conditional probabilities (such as the relation between symptoms and diseases), or conditional sentences (such as, "Birds fly," "Fire causes smoke" and

"Smoke suggests fire.") cannot be properly encoded in the random-assumptions framework, until we have sufficient information to form a complete probability model. Since conditional sentences make up the bulk of human knowledge, this limitation essentially means that domain knowledge as we know it is not SA-encodable.

The prevailing practice in the design of SA-TMS systems (e.g., Laskey and Lehner 1989) has been to represent the rule "If A then B" by the material implication formula $A \supset B = (\neg A \vee B)$ and assign to this formula some weight w that measures the strength of the rule or its validity, thus converting the rule into a bona fide belief function satisfying $m(A \supset B) = w$. This practice is not entirely without merit. For example, combining the resulting belief function with the evidence $A = true$ does give the expected result $Bel(B \mid A) = w$. Moreover, if we are given a full specification of a joint probability, we can replace every conditional probability by its material implication counterpart, combine these functions using Dempster's rule, and the result would be equivalent to the original probability model. The problem begins when the probabilistic model is incomplete and some of the conditional probabilities (or the priors) are missing. In such cases, the material implication scheme may yield very undesirable effects, examples of which are shown next.

Example 5 (Chaining): Consider the following two rules:

r_1: If the ground is wet, then it rained last night (m_1),

r_2: If the sprinkler was on, then the ground is wet (m_2).

If we find that the ground is wet, rule r_1 tells us that $Bel(Rain) = m_1$. Now, suppose we learn that the sprinkler was on. Instead of decreasing $Bel(Rain)$ by explaining away the wet ground, the new evidence leaves $Bel(Rain)$ the same. More seriously, suppose we first observe the sprinkler. Rule r_2 will correctly predict that the ground will get wet, and without even inspecting the ground, r_1 will conclude that it rained last night, with $Bel(Rain) = m_1 m_2$.

Rule chaining can be especially bothersome when combined with contraposition, $(a \rightarrow b) \Rightarrow (\neg b \rightarrow \neg a)$, another feature inherent to the material implication.

Example 6 (Contraposing): Consider the rules:

If a person is kind, then that person is popular (m)
If a person is fat, then that person is unpopular (m)

Learning that Joe is fat produces the strange result that Joe is believed to be unkind with strength m^2.

The last two examples represent difficult challenges to any logic that sanctions indiscriminate contraposition, oblivious to the direction of causation (Hanks and McDermott 1986). In the probability bounds approach causation is encoded as specifications of conditional independence relations, usually in graphical forms [Pearl 1987, 1988]. The sampled assumptions approach cannot admit such specifications when rules are encoded as randomized material implications, because the latter are invariant to contraposition: $Bel\ (A \supset B) = Bel\ (\neg B \supset \neg A)$.

Example 7 (Reasoning by Cases): Suppose we are given the following two rules:

If A then B, with certainty 0.8

If $\neg A$ then B, with certainty 0.7.

Common sense dictates that even if we do not have any information about A we should still believe in B to a degree at least 0.7. The sampled-assumptions approach does not support this intuition. If we try to encode the rules as material implication formulae, sampled according to:

$$m_1(A \supset B) = 0.8 \qquad m_2(\neg A \supset B) = 0.7$$

$$m_1(True) = 0.2 \qquad m_2(True) = 0.3$$

and combined by Dempster's rule, we obtain $Bel(B) = 0.56$, in clear violation of common sense.

Example 8 (Specificity): Consider the following set of rules:

Rule-1: Typically penguins do not fly
Rule-2: Penguins are birds
Rule-3: Typically birds fly

Suppose we know that Tweety is both a penguin and a bird, and we wish to assess the belief that Tweety flies. Any assessment method based on sampling these rules as independent Boolean assumptions will remain oblivious to the second rule, stating that all penguins are birds, because knowing that Tweety is both a penguin and a bird subsumes the information provided by that rule. Thus, the computed value of $Bel(Tweety\ flies)$ will be solely a function of the weights assigned to Rule-1 and Rule-3, regardless of whether penguins are a subclass of birds or birds are a subclass of penguins. This stands contrary to common discourse, where people expect class properties to be overridden by properties of more specific subclasses. By comparison, the probability-bound approach does yield the expected results (i.e., that Tweety most likely does not fly) if the rules are treated as conditional probability specifications, infinitesimally close to 1 [Pearl 1988, 1989].

6. Conclusions

We have described two paradigms that deal with incomplete specifications of probabilistic knowledge; one based on probability-bounds, and the other on sampled-assumptions. The former treats the specifications as hard constraints over probabilities and computes the highest level that can provably be attached to the probability of a query. The latter treats the specifications as instructions for sampling and adopting assumptions and, after examining their logical consequences, it computes the probability that a query is provable.

We have identified and exemplified two major shortcomings of the sampled-assumptions approach. First, the failure to represent independencies among events with unknown probabilities. This leads to peculiar behavior in applications such as circuit diagnosis, where the computed beliefs stand contrary to the available information, and might lead to unreasonable decisions and test strategies. Second, the failure to represent domain knowledge cast in the form of defeasible conditional sentences. This limits the applications of sampled-assumptions techniques to cases where domain knowledge is articulated in purely categorical terms. These include, for example, strict taxonomic hierarchies, terminological definitions and descriptions of deterministic systems (electronic circuits), but exclude domains in which the rules tolerate exceptions (e.g., medical diagnosis and default reasoning).

Future studies should determine whether there are restricted forms of knowledge representation that are amenable to sampled-assumptions strategies, safe from the paradoxes uncovered in this paper.

References

[Aleliunas 1988] Aleliunas, R. A. 1988. A new normative theory of probabilistic logic. *Proc., 7th Biennial Conf. of the Canadian Society for Computational Studies of* Intelligence (CSCSI-88), Edmonton, Alta., 67-74.

[de Finetti 1974] de Finetti, B. 1974. *Theory of probability.* New York: Wiley.

[de Kleer 1986] de Kleer, J. 1986. An assumption-based truth maintenance system. *Artificial Intelligence* 29:241-88.

[Dempster 1967] Dempster, A. P. 1967. Upper and lower probabilities induced by a multivalued mapping. *Ann. Math. Statistics* 38:325-39.

[Hanks and McDermott 1980] Hanks, S., and McDermott, D. V. 1986. Default reasoning, nonmonotonic logics, and the frame problem. *Proc., 5th Natl. Conf. on AI (AAAI-86)*, Philadelphia, 328-33.

[Good 1950] Good, I. J. 1950. *Probability and the weighing of evidence.* London: Griffin.

[Laskey and Lehner 1989] Laskey, K.B. and Lehner, P.E. 1989. Assumptions, beliefs and probabilities. *Artificial Intelligence* 41(no. 1): 65-77, November.

[Nilsson 1986] Nilsson, N. J. 1986. Probabilistic logic. *Artificial Intelligence* 28 (no. 1):71-87.

[Pearl 1987] Pearl, J. 1988b. Embracing causality in formal reasoning. *Artificial Intelligence* 35 (no. 2):259-71.

[Pearl 1988] Pearl, J. 1988. *Probabilistic reasoning in intelligent systems.* San Mateo: Morgan Kaufmann.

[Pearl 1989] Pearl, J. 1989. Probabilistic semantics for nonmonotonic reasoning: A survey. *Proc., First Intl. Conf. on Knowledge Representation and Reasoning*, Toronto, Canada, 505-516.

[Provan 1989] Provan, G.M. 1989. A logical interpretation of the Dempster-Shafer theory, with application to visual recognition. *Proc., 5th Workshop on Uncertainty in AI*, Windsor, Ontario, Canada, 287-299.

[Shafer 1976] Shafer, G. 1976. *A mathematical theory of evidence.* Princeton: Princeton University Press.

[Smith 1961] Smith, C. A. B. 1961. Consistency in statistical inference and decision. *J. Royal Statist. Soc.*, ser. B 23:218-58.

A Study of Conflicting Evidence in Probabilistic Logic

Dr. Mary McLeish*
University of Guelph
Guelph, Ontario N1G 2W1

Abstract

A problem for practitioners in using probabilistic logic (N. Nilsson) has been the model's inability to deal with an inconsistent assignment of probability values in the valuation vectors. As noted by Nilsson, probabilistic logic is inherently monotonic - one can only reduce the region of consistent valuations when adding new information. This paper presents a solution which modifies the original model. Conditions are provided which show when results can be obtained. The ϵ-calculus for defaults introduced by J. Pearl is compared with an extension to probabilistic logic and the results are found to be comparable. Computationally feasible solutions to problems of conflicting evidence and non-monotonicity in probabilistic logic are presented.

1 Introduction

In commonsense reasoning, it is often the case that one wishes to draw 'typically' information from a fact such as 'usually all A's are B's' or most A's are B's. As some exceptions to these rules do exist, they may be at best taken to be 'default' rules (Hanks and McDermott [8],Reiter [17]) or assumptions. Under the default assumption, the rules are considered to be true until this belief needs to be revised due to the arrival of contradictory pieces of evidence (such as A is definitely not B, for some A.) Hence the inherently non-monotonic nature of default reasoning systems. In the original scheme proposed by Reiter, no use is made of multivalued logic, probability theory or any (quasi) numeric form of uncertainty. Many recent papers have appeared (Ginsberg [6], Grosof [7], Kadie [9], Yager [18]) to name but a few discussing the problem in a variety of these frameworks (which should also include the fuzzy approach, Zadeh [19]).

Probabilistic Logic has been discussed by Nils Nilsson in [15], as a way of entailing inexact sentences. It is inherently a monotonic reasoning system as inconsistencies are not allowed to take place as part of the nature of the logic (see final remarks in [15]). In practice however, experts may often assign beliefs which are inconsistent in Nilsson's framework. Sometimes attempts are then made to artificially adjust these values to produce a consistent set, but this is a very ad hoc and often impossible task. In earlier papers by this author, the basic entailment scheme is examined for cases where A holds, $A \rightarrow B$, but B does not necessarily hold (c.f. [11,12,13]). Non-monotonicity was accomplished by the addition of inconsistent valuation vectors which were not assigned a zero probability value. In other words, it was assumed that there was a possible 'inconsistent' world. The earlier papers examined the entailment results in this situation. In [13], a connection between Dempster-Shafer theory and probabilistic logic is used to derive entailment values for the max entropy solution which are more meaningful than that first found in [11] and [12].

J. Pearl in [16] and other papers, suggests the use of an epsilon calculus to capture the idea of non-monotonicity. In particular, if one has a set of ϵ-bound conditional probability interpretations of the form $\{P(f/b) \geq 1-\epsilon$, $P(f/p) \leq \epsilon$ and $P(b/p) \geq 1-\epsilon\}$, one would like the 'plausible' conclusion of a theory with these defaults to find that $P(f/p,b) \geq 1-0(\epsilon)$, where $0(\epsilon)$ stands for any function of epsilon whose limit as $\epsilon \rightarrow 0$ is also zero. (Here the common interpretation of f, p and b are flies, penguin and bird). A detailed theory is developed which will guarantee that only 'plausible' conclusions are found. This paper examines how probabilistic logic deals with such a scenario and whether or not it can draw 'plausible' conclusions. Some general results are presented indicating when solutions can be obtained and the nature of the entailed values. The model which adds new vectors is shown to be equivalent to a method of variable splitting. Both models are very intuitive and lead to solutions compatible with those in [16] by J. Pearl.

These new results are all easily computable and provide a feasible solution to the problem of non-monotonicity and conflicting evidence in probabilistic logic.

Section 2 presents the necessary background information and terminology. Section 3 introduces the notion

*This work has been supported by NSERC operating grant #A4515.

of extreme probabilities to probabilistic logic and compares the addition of inconsistent vectors to a method of variable splitting through the study of several examples. Section 4 presents some general results (theorems) providing conditions for the existence of solutions in the presence of inconsistent belief assignments. Section 5 concludes the paper.

2 Probabilistic Logic and Defaults

Notation from Probabilistic Logic and a Brief Discussion of the ϵ-calculus approach

Probabilistic Logic is essentially an extension of first-order logic in which truth values of sentences can range between 0 and 1. A short summary of the scheme is presented below.

S represents a finite sequence L of sentences arranged in arbitrary order, e.g. $S = \{S_1, S_2, \ldots, S_L\}$.

$V' = \{v_1, v_2, \ldots, v_L\}$ is a valuation vector for S, where $'$ denotes transpose and $v_1 = 1$ if S_i has value true, $= 0$ otherwise.

V is consistent if it corresponds to a consistent valuation of the sentences of S. v is the set of all consistent valuation vectors for S and let $K = ||v||$ (cardinality). (Note $K \leq 2^L$). Each consistent V corresponds to an equivalence class of "possible worlds" in which the sentences in S are true or false according to the components of V. Let M (sentence matrix) be the $L \times K$ matrix whose columns are the vectors in v. Its rows will be denoted by S. If P_i is the i'th unit column vector, $MP_i = V_i$, where V_i is the ith vector of v.

Example: Let

$$S = (A, A \supset B, B) \; v = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \text{ and}$$

$$M = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

However, if each of the sentences' truth values are uncertain in some sense, a probability distribution over classes of possible worlds is introduced.
$P' = \{P_1, P_2, \ldots, P_k\}$ with $0 \leq P_i \leq 1$ and $\sum_i P_i = 1$.

Here the i'th component of P represents the probability that 'our' world, is a member of the i'th class of worlds. Now a consistent probabilistic valuation vector V over the sentences in S is computed from the equation $V = MP$. The components of V are the probabilities of the S_i being true (or the probability that 'our' world is a member of one of those classes of possible worlds in which sentence S_i is true).

Returning to example 1, we find that even if consistent valuations are known for the sentences A and $A \supset B$, the probability of B is not necessarily uniquely defined. One can determine the bounds $P(A \supset B) + P(A) - 1 \leq P(B) \leq P(A \supset B)$, which provide some restrictions. However, often a more precise value for $P(B)$ needs to be predicted. A method using a maximum entropy approach (borrowed from P. Cheeseman [4]) is used to obtain an exact solution for $P(B)$. We assume $P(A \supset B) = q$, $P(A) = p$ and add the constraint, $\sum_i P_i = 1$ by letting the first row of the matrix M be all 1's and the first entry of the valuation vector be 1. (see 15 for further details).

The entropy function becomes
$H = -P.\log P + l_1(v_1 - S_1.P) + l_2(v_2 - S_2.P) + \ldots l_L(v_L - S_L.P)$, where the l_i are Lagrange multipliers. Following Cheeseman, the solution for maximum entropy becomes

$$P_i = e^{-1}.e^{-l_1 S_{i1}} \ldots e^{-l_L S_{Li}}$$

If one employs this method, at least for example 1, the solution for $P' = \{p + q - 1, 1 - q, (1 - p)/2, (1 - p)/2\}$, when $V' = \{1, p, q\}$ and thus $P(B) = p/2 + q - \frac{1}{2}$.

Another 'projection' method has been suggested by Nilsson [15]. This author has shown in [14] under which conditions these solutions are the same. It is mainly the maximum entropy approach that is used in this paper.

One problem with this approach concerns the range of values for which the entailment scheme will work. Clearly $p + q$ must be greater than or equal to 1 for the solution vector to be meaningful. For the entailment result to be a valid one ((i.e.), in the range $0 \leq P(B) \leq 1$, then $p + 2q \geq 1$. Therefore, $p + q \geq 1$ is a sufficient condition for both. However the values for p and q might be expert defined (obtained from expert opinions) and not satisfy this condition. If interpreted as conditional probabilities, statements like $A \supset B$ become $P(B/A)$. The frequentist interpretation of the conditional probability will produce values of $P(A)$ and $P(B/A)$ which need not obey any condition like $P(A) + P(B/A) \geq 1$. Of course $P(A \supset B)$ is more like $P(\sim A) + P(B/A)$, but this is not correct either as this quantity could be greater than 1.

One should be considering $P(\sim A \vee B)$, although, in practise, it is often easier to find $P(B/A)$ than the prior of $\sim A \vee B$. (See Kane [10] where a further discussion is made of using $P(B/A)$ to represent $A \supset B$.) For example, if we are considering the chance that birds fly, it is easier to estimate P(flies/bird) than $P(\sim$ bird or flies) from a population of all living things. Thus even with a frequency interpretation, if these values are being estimated (from sample populations), the results may not give you $p+q \geq 1$ (although their true values from the total population should).Thus observed values of 0.4 and 0.5, for example would lead to an ill-defined P vector (the first element would have value -0.1). An entailment value of 0.2 for $P(B)$ can be calculated, but it is questionable what this means. If $p = 0.5$, $q = 0.2$, then $P(B) = -0.5$, again a meaningless result.

Thus, there are two ways in which results can be invalid:

(i) the solution vector has a negative component, but a positive value for $P(B)$ is obtained.

(ii) both a component of the position vector and $P(B)$ are negative.

These invalid situations arise when the probability values lie outside the consistent region. The question remains, however, that this system cannot be used to predict values for $P(B)$, when $p+q<1$, which can occur for perfectly reasonable and possibly observed values of p and q.

In the papers [12] and [13] by this author, Nilsson's work has been reconsidered introducing an inconsistent vector to his M matrix. This was done primarily to represent default reasoning and subsequent non-monotonicities that can occur. In [13], a relationship to Dempster-Shafer theory is presented which helps overcome the problems of (i) and (ii). These difficulties have been discussed here because they also arise when considering the use of probabilistic logic for very small and large probabilities and it is really these problems that again force the introduction of inconsistent vectors.

The ϵ-calculus formulation in [16] presents a way in which to capture the essence of statements like "almost all A's are B's" by interpreting these sentences in the terms of extreme conditional probabilities, infinitesimally distant from 0 or 1. For example, the sentence almost every bird flies becomes $P(\text{Fly}(x)/\text{Bird}(x)) \geq 1-\epsilon$, where $\epsilon > 0$ is a quantity that can be made infinitesimally small. A default theory $T = < F, \Delta >$ contains factual sentences F and default statements Δ. The notion of a plausible conclusion of a theory is given as follows: let $\Delta_\epsilon = \{P(q/p) \geq 1-\epsilon : p \rightarrow q \epsilon \Delta\}$. Then r is a plausible conclusion of T if $P(r/F) \geq 1-0(\epsilon)$ where $0(\epsilon)$ has been described in the introduction. A full theory is developed tying the ϵ-theory together with the work of E. Adams on the Logic of Conditionals.

Pearl examines the results of the analysis of the example of drawing a plausible conclusion from:

$$\begin{aligned} \Delta &= \{\text{Penguin} \rightarrow \text{Fly}, \text{Bird} \rightarrow \\ &\quad \text{Fly and Penguin} \rightarrow \text{Bird}\}, \\ \text{and} \quad F &= \{\text{Penguin(Tweety)}, \text{Bird(Tweety)}\} \end{aligned}$$

If $\Delta_\epsilon = \{P(f/b) \geq 1-\epsilon, P(f/p) \leq \epsilon \text{ and } P(b/p) \geq 1-\epsilon\}$ it is shown in [16], following the usual rules of probability that $P(f/p,b) \leq 0(\epsilon)$ or equivalently that $P(\sim f/p,b) \geq 1-0(\epsilon)$, where $0(\epsilon) = \frac{\epsilon}{1-\epsilon}$. Thus, Tweety does not fly is a plausible conclusion of the theory. This interpretation of the classical example allows for bird $\rightarrow$ fly and penguin $\rightarrow\sim$ fly to both hold - permitting non-monotonicity. The next section will show how probabilistic logic can also capture the same result.

3 Probabilistic Logic and Extreme Probabilities

There are several ways of representing the casuality just presented in probabilistic logic. The most general information is presented below. Here V_2 would be $1-\epsilon_1$, V_4 is $1-\epsilon_2$, V_5 is ϵ_3, where ϵ_1, ϵ_2 and ϵ_3 are all bounded above by some small quantity ϵ. (The different ϵ_i are used to facilitate computations in terms of equalities involving the ϵ_i, instead of inequalities involving ϵ.) Now to be very general, V_1 and V_3 could be taken to be less than 1. If they are identically 1, the matrix should be automatically reduced, as will be discussed later. This matrix relation presents the consistent valuation vectors associated with the entailment scheme corresponding to the casuality sought in the example in Pearl [16]. The notation follows that outlined in §2.

$$\begin{array}{c} b \\ p \rightarrow b \\ p \\ b \rightarrow f \\ p \rightarrow f \\ f \end{array} \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} P_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ P_8 \end{pmatrix} = \begin{pmatrix} V_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ V_6 \end{pmatrix} \quad (1)$$

Since the vector of 1's associated with $f, S(f)$, is a subset of that for $p \rightarrow f$ (i.e. $\exists 1$ in the jth place for $S(f) \rightarrow \exists 1$ in the jth place for $S(p \rightarrow f)$), the probability of f will be less than or equal to that of $p \rightarrow f$. That is $V_6 \leq V_5$, where $0 \leq P \leq 1$.

Therefore, if V_5 is known to be $\leq \epsilon$, so will V_6 be $\leq \epsilon$. This would seem to agree with the ϵ-calculus result.

However, on closer examination of the system one can prove that no proper solution to the resulting set of equations exists. The P_i's obtained from max entropy are not between 0 and 1 and indeed a careful examination of the bounds on the P_i imposed by the resulting system shows a solution giving a proper probability distribution for the P_i to be impossible under the constraints that $V_4 \geq 1-\epsilon$ and $V_5 \leq \epsilon$. (The situation is similar to the problem discussed in 2.1).

One can examine a number of different situations of interpretations for the values of V_1, V_2, V_3, but it will always be considered that V_2 is close to 1 and so is V_3 (to represent the ϵ-calculus example). When this is required, probabilistic logic finds the system insolvable for the P_i, assuming $\sum P_i = 1$ and $0 \leq P_i \leq 1$. In essence, it is saying there is something incompatible about the assignment of the V_i values and, in some sense, this is a reasonable conclusion. That is, in its usual interpretation, probabilistic logic cannot deal with the inherent non-monotonicity of the probability (or belief) values assigned to the V_i's, although such an assignment might well occur in commonsense reasoning.

The previous discussion can be stated more formally in the following theorem.

Theorem 1

The entailment scheme representing the sentences used in the example in Pearl [16] shown by the matrix equation 1 does not have any solution for the P_i satisfying $1 \leq P_i \leq 1$ and $\sum P_i = 1$ under the assumption that ϵ is a small quantity.

Proof: Under row reduction the system reduces to the set of equations:

$$\begin{array}{lc}
1) & P_3 + P_4 = 1 - \epsilon_5 \\
2) & P_1 + P_3 + P_5 + P_7 = 1 - \epsilon_1 \\
3) & P_6 + P_8 = \epsilon_1 - \epsilon_2 \\
4) & P_1 + P_2 = \epsilon_5 - \epsilon_3 \\
5) & P_4 + P_5 + P_6 + P_8 = 1 - \epsilon_4 - \epsilon_5 + \epsilon_3 \\
6) & P_5 + P_6 + P_7 + P_8 = \epsilon_3
\end{array}$$

Here V_1 to V_5 are $1-\epsilon_1, \ldots 1-\epsilon_4$ and ϵ_5 respectively with the ϵ_i's bounded above by some small quantity ϵ. (A new first row of the matrix from e.g. (1) consisting of all 1's has been added to the system as in the discussion in 2.1). Now $P_5+P_6+P_8 \le \epsilon_3$ from (6). Therefore $P_4 \ge 1-\epsilon_4-\epsilon_5$ from (5). From (1), $P_3 \le \epsilon_4$. From (4), $P_1 \le \epsilon_5 - \epsilon_3$. Therefore $P_1+P_3+P_5+P_7 \le \epsilon_5-\epsilon_3+\epsilon_4+\epsilon_3 = \epsilon_5+\epsilon_4 \le 2\epsilon$. On the other hand this is supposed to equal $1-\epsilon_1$ from equation 2. Thus, unless $\epsilon \ge 1/3$, no solution is possible.

The same result could have been arrived at by considering the smaller system obtained when p is true. The matrix becomes

$$\begin{array}{c} p \\ p \rightarrow b \\ b \rightarrow f \\ p \rightarrow f \\ f \end{array} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ P_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1-\epsilon_1 \\ 1-\epsilon_2 \\ \epsilon_3 \end{pmatrix}.$$

This system has a unique solution with $P_2 = \epsilon_1 + \epsilon_2 + \epsilon_3 - 1$, which implies some $\epsilon_i \ge 1/3$ if $0 \le P_4 < 1$, which contradicts ϵ_i being an infinitesimally small quantity.

It is with this smaller system that a natural mechanism for obtaining a solution will be sought. The underlying reason that no solution can be found stems from the lack of any column vector (world) which represents the situation evident from the given valuation vector; namely the world $(1,1,1,0)$ in which p is true, $p \rightarrow b$ is true, $b \rightarrow f$ is true but $p \rightarrow f$ is false. The next section will describe ways of viewing the addition of this vector in both logically consistent and inconsistent fashions and the results of doing this.

Variable Splitting and the Use of Inconsistent Vectors

One method of obtaining a viable solution is through a separation of the two opposing conclusions for f. A mechanism for doing this produces the following matrix equation, where f has been split into two variables f_1 and f_2, which are then recombined to represent finding the probability that $f_1 = f_2 =$ true.

Example 1

$$\begin{array}{c} p \\ p \rightarrow b \\ b \rightarrow f_1 \\ p \rightarrow f_2 \\ f_1 \cap f_2 \end{array} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ P_7 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ 1-\epsilon_1 \\ 1-\epsilon_2 \\ \epsilon_3 \\ P(f_1 \cap f_2) \end{pmatrix}$$

Columns 2 and 3 can be collapsed to one (column 3) as the max entropy solution will assign $P_2 = P_3$. (These values can be split apart later. See Kane [10] for a reference which combines duplicate vectors in the max entropy solutions to probabilistic logic.) The matrix (without the last row) row reduces to

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ P_6 \end{pmatrix} = \begin{pmatrix} \epsilon_2 \\ 1-\epsilon_1-\epsilon_2 \\ \epsilon_1 \\ \epsilon_3 \end{pmatrix}$$

The maximum entropy solution becomes:

$P_1 = \epsilon_3(1-\epsilon_1-\epsilon_2)$, $P_2 = \epsilon_1\epsilon_3$, $P_3 = \epsilon_2(1-\epsilon_3)$, $P_4 = \epsilon_2\epsilon_3$, $P_5 = \epsilon_1(1-\epsilon_3)$, $P_6 = (1-\epsilon_1-\epsilon_2)(1-\epsilon_3)$. If we split P_2, we have $P_2 = \frac{\epsilon_1\epsilon_3}{2}$, $P_3 = \frac{\epsilon_1\epsilon_3}{2}$ and $P_i = P_{i+1}$, $i \ge 4$. Then $P(f_1 \cap f_2) = \epsilon_3(1-\epsilon_2)$ or $\epsilon_3(1-\frac{\epsilon_1}{2}-\epsilon_2)$ for the more complete solution with 7 column vectors in M.

The key vector which has been added to the original system is the vector $\begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$. In the original system where f is not split into two variables, this vector would be inconsistent. However it is most representative of the given values in the valuation vector. That is $\begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$ is a world that is likely to have the highest probability given the valuation $\begin{pmatrix} 1 \\ 1-\epsilon_1 \\ 1-\epsilon_2 \\ \epsilon_3 \end{pmatrix}$.

One may obtain an approximate solution to the original system by the addition of this inconsistent vector alone: (i.e.) one obtains

Example 2

$M = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$. As the worlds which were included in the larger matrix where f was replaced by f_1 and f_2 have small probability of occurrence, the results will not be very different. The max entropy solution is: $P_1 = \frac{\epsilon_3(1-\epsilon_1-\epsilon_2)}{1-\epsilon_2}$, $P_2 = \frac{\epsilon_1\epsilon_3}{1-\epsilon_2}$, $P_3 = \epsilon_2$, $P_4 = \frac{\epsilon_1(1-\epsilon_2-\epsilon_3)}{1-\epsilon_2}$, $P_5 = \frac{(1-\epsilon_1-\epsilon_2)(1-\epsilon_2-\epsilon_3)}{1-\epsilon_2}$; all the P_i obey $0 \le P_i \le 1$. $P(f) = \epsilon_3$.

In both examples, $P(f) \le \epsilon_3$ and is $0(\epsilon_3)$, consistent with the probabilistic interpretation in Pearl [16]. In both cases, P_5 (or P_7) are $0(1)$, (ie) approach 1 as $\epsilon_i \to 0$ and all other P_i are $0(\epsilon_i')$, where ϵ_i' is a function of the ϵ_i and $\epsilon_i' \to 0$ as $\epsilon_i \to 0$.

In fact, one can easily find a particular solution of the underconstrained set of equations representing the system M with the addition of one inconsistent vector: namely $P_1 = \epsilon_3$, $P_2 = 0$, $P_3 = \epsilon_2$, $P_4 = \epsilon_1$, $P_5 = 1 - \epsilon_1 - \epsilon_2 - \epsilon_3$. A solution to the homogeneous system is $P(\text{transpose}) = (1, -1, 0, 1, -1)$. Thus the general solution [see 14] is $(\epsilon_3, 0, \epsilon_2, \epsilon_1, 1 - \epsilon_1 - \epsilon_2 - \epsilon_3) + k(1, -1, 0, 1, -1)$. In order for the P_i to remain constrained by $0 \le P_i \le 1$, k must be negative as $P_2 = 0 + k(-1)$. From $P_5 = 1 - \epsilon_1 - \epsilon_2 - \epsilon_3 - k \le 1$, let $j = -k$, (ie) $|k|$ then $j \le \epsilon_1 + \epsilon_2 + \epsilon_3$ and so $|k| = 0(\epsilon')$, $\epsilon' = \epsilon_1 + \epsilon_2 + \epsilon_3$. This means that for any solution, all the P_i will be small ($0(\epsilon')$, for some ϵ' a function of the ϵ_i which goes to 0 as $\epsilon_i \to 0$), except P_5 which will be $0(1)$. (The value of k which gives the maximum entropy solution is $\frac{-\epsilon_1\epsilon_3}{1-\epsilon_2}$.)

The next section will state some general results which show that it is only by the addition of vectors like the inconsistent vector added in Example 2 to the original system which make a solution possible.

4 General Results for Conflicting Evidence

Theorem 2

For any set of sentences corresponding to a valuation vector of the form $V = \{1, 1 - \epsilon_1, \ldots 1 - \epsilon_j, \epsilon_{j+1}, \ldots \epsilon_N\}$, a proper solution to the matrix equation $MP = V$ will exist only if there exists at least one column vector in M of the form $(\underbrace{1, 1, 1, \ldots 1}_{j+1}, \underbrace{0, 0, \ldots, 0}_{N-j})$.

Here it will be assumed M is an N by L matrix with a first row of all ones. P stands for a particular solution vector of P_i's. A proper solution means a solution for which $0 \le P_i \le 1, \forall i = 1, \ldots L$. A lemma will first be proven from which the necessity of the string of 1's will follow.

Lemma 2.1: There is a unique minimal subset of the P_i, $\{P_{i_m}\}$, such that $\sum_{i=1}^{t} P_{i_m} = 1 - 0(\epsilon)$.

Proof:

Consider any row R of the matrix M corresponding to a valuation of order $1 - \epsilon$. Then the inner product of this row with P equals $\sum_{i=1}^{q} P_{i_m} = 1 - 0(\epsilon)$ where q is the number of non-zero entries of R. Remove all P_{i_m} such that $0(P_{i_m}) = \epsilon$ and renumber the P_{i_m}'s such that we now have $\sum_{i=1}^{t} P_{i_m} = 1 - 0(\epsilon)$. Now none of these remaining P_{i_m} will be of order ϵ. Suppose there exists another set P_{i_k} such that $\sum_{i=1}^{w} P_{i_k}$ is $1 - 0(\epsilon)$. If it is disjoint from the P_{i_m}'s then $\sum_{i=1}^{L} P_i$ will be $2 - 0(\epsilon)$ instead of 1. If the two sets intersect, suppose their intersection is say $\{P_{i_\ell}\}$, $\ell = 1, \ldots r$. Then $\sum_{i=1}^{L} P_i = 2 - O(\epsilon) - \sum_{\ell=1}^{r} P_{i_\ell} = 1$ implies $\sum_{\ell=1}^{r} P_{i_\ell} = 1 - O(\epsilon)$, and the two other sets were not minimal - unless one of them equalled $\{P_{i_\ell}\}$.

Proof of Main Theorem:

The fact that there must exist at least one column vector in M whose $j + 1$st entries are all 1 follows from Lemma 2.1 which ensures that each row corresponding to a valuation of $1 - 0(\epsilon)$ must contain a 1 in column positions i_m.

To complete the proof, suppose some entry of the columns $i_1 \ldots i_t$ in rows $j + 2$ through N is non-zero. Then the inner product of that row with P will result in an non-infinitesimal value, which contradicts the valuation value of $0(\epsilon)$. Another way of looking at it is to note that if the valuation is $0(\epsilon)$, then some P_{i_m} is $0(\epsilon)$ which contradicts the minimality of the set $\{P_{i_m}\}$. Therefore there exists at least one column vector of the form $(\underbrace{1, 1, \ldots 1}_{j+1}, \underbrace{0, 0 \ldots 0}_{N-j})$. The following Corollaries follow immediately from this theorem.

Corollary 2.1: Given the equation $MP = V$ with a valuation vector of the type described by Theorem 2, if a proper solution exists it will be of the form $P_j \le \epsilon$ for all $j \ne i_m$, $m = 1, \ldots t$ and the P_{i_m} sum to $1 - 0(\epsilon)$ and are not infinitesimal.

Considering the entailment examples under study, there will usually be only one vector of the form $(1, 1, \ldots 1, 0, 0, 0, \ldots)$ in the system and it will have a final value of 0 when the conclusion row is added. Suppose this is the L'th column vector. Then one may state the following more precise result:

Corollary 2.2: If there is only one column vector of the form $(\underbrace{1, 1, \ldots 1}_{j+1}, \underbrace{0, 0 \ldots 0}_{N-j})$ in M, then any proper solution that exists will be of the form

$$P_L = 1 - 0(\epsilon) \text{ and } P_i = 0(\epsilon) \forall i = 1, \ldots L - 1.$$

It follows immediately from this that if the Lth entry of the $N + 1$st row (conclusion) is 0, then the probability of the entailed result will be $0(\epsilon)$ as it will equal the sum of a subset of the P_i, where $i \le L - 1$.

Corollary 2.3: If the Lth entry of the $N + 1$st row (entailment result) is 0, the probability or belief in the entailment will be $0(\epsilon)$.

The full converse to Theorem 2 is harder to prove rigorously without a more precise knowledge of M. However, if even one vector of the form required by Theorem 2 is present in M, say in the Lth column, usually a solution will be found by setting $P_L = 1 - f(\epsilon)$, for some function of the ϵ_i of order ϵ, and allowing the other P_i to be $0(\epsilon)$. If the system is underconstrained, solutions should exist for the resulting equations and thus a solution with the required bounds on the P_i.

As we are primarily concerned with problems of conflicting evidence, which might arise in a variety of ways, let us study a few examples of what might closely represent the final results after a chain of some unknown sentences.

Example 3: Suppose we have a series of entailed sentences which eventually result in implications of the form $a \rightarrow f$ and $b \rightarrow f$ with opposing extreme valuations. Assume a and b are effectively true. (Essentially the same results are obtained if they are taken to have truth value $1 - 0(\epsilon)$). Using the method of separation of variables, we have (recalling that the first row of 1's represents the constraint that the P_i's sum to 1):

$$\begin{array}{c} \\ a \rightarrow f_1 \\ b \rightarrow f_2 \\ f_1 \cap f_2 \end{array} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1-\epsilon_1 \\ \epsilon_2 \end{pmatrix}$$

One can easily determine the max entropy solution: $P_1 = \epsilon_2(1-\epsilon_1)$, $P_2 = \epsilon_1(1-\epsilon_2)$, $P_3 = (1-\epsilon_2)(1-\epsilon_1)$ and $P_4 = \epsilon_1\epsilon_2$.

The entailed $P(f_1 \cap f_2) = \epsilon_2(1-\epsilon_1)$, which is consistent with that found from the larger system used in example 1. As illustrated with example 2, we could have simply added the inconsistent vector $(1,0,0)$ to the system $a \rightarrow f$, $b \rightarrow f$ and obtained equivalent results. (ie)

Example 4:

$$\begin{array}{c} \\ a \rightarrow f \\ b \rightarrow f \\ f \end{array} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1-\epsilon_1 \\ \epsilon_2 \end{pmatrix}$$

Now there is a unique solution, $P_1 = \epsilon_2$, $P_2 = \epsilon_1$, $P_3 = 1 - \epsilon_1 - \epsilon_2$ and $P(f) = \epsilon_2$.

One notes that the probability of the dropped world, (0 1 0), in going from example 3 to example 4 is $\epsilon_1\epsilon_2$ or extremely small. This was also the case in the expanded examples 1 and 2 where $P_4 = \epsilon_2\epsilon_3$ is the probability of the dropped vector.

Another interpretation might be to consider $f_1 \cup f_2$ as a conclusion. However, usually one wishes to 'and' the two rules together. This conclusion would be more like taking one or the other. Of course what is desired depends on the problem begin modelled. In the 'Tweety' example, as $p \rightarrow b$ it would seem that one really wants to test $f_1 = f_2 =$ True, which can be represented by 1's in the $f_1 \cap f_2$ vector.

Without considering the connection between adding an inconsistent vector and variable splitting, one might be tempted to add the inconsistent vector twice: once with outcome 1 and once with outcome 0. This results in final entailed values within some ϵ of 1/2 (more like what would be obtained using the MYCIN model of certainty factors [2]). However, I reject this interpretation as it is not consistent with the model or result of using the two variables f_1 and f_2 and has no meaning in that context. Only if the users of the system had some preference for wishing these types of conclusion from conflicting evidence should they be used instead of what has been considered in this paper. The results reflecting the $f_1 \cap f_2$ conclusion are the more natural and logical conclusions from probabilistic logic.

5 Conclusions

In order to interpret the problem of conflicting evidence (or rules implying conflicting results) in probabilistic logic, either additional variables need to be added to the system or else the inconsistent column vector which most closely represents the valuation vector V needs to be added to the system. It is only then that a proper solution can be found. The probability the world is in a state represented by the inconsistent vector will be $1 - 0(\epsilon)$ and the weight on the other worlds will be $0(\epsilon)$. If the value under the added column vector in the conclusion row is 0, P(conclusion being true) will be $0(\epsilon)$. The examples show that adding the inconsistent vector is essentially equivalent to the method of variable splitting. It should also be noted that a number of variations on the representation were tried with essentially the same results (ie) changing the conclusion to $\sim f$, or using $p \rightarrow \sim f$ now with valuation $1 - \epsilon$ resulted in the same problems of conflict (no solution) or essentially the same solutions with the introduction of the appropriate inconsistent vector.

Studying the examples 1 and 2, we see that the results are consistent with those found by Pearl [16] for the non-monotonic problem. Examples 3 and 4 are representative of many entailment schemes. This paper has shown how to deal with these problems in the framework of probabilistic logic. The necessity of the addition of an inconsistent 'world' which closely resembles the conflicting evidence in the valuation vector is intuitively obvious and has been shown mathematically to be the only way to obtain a solution.

References

[1] Bundy, A. "Incidence Calculus: A Mechanism for Probabilistic Reasoning", *Workshop on Uncertainty and Probability in A.I. Proceedings*, (1985), pp. 177-84.

[2] Buchanan, B.G., Shortliffe, E.H. *Rule-Based Expert Systems*, Addison-Wesley, (May, 1985).

[3] Cheeseman, P. "A Method of Computing Generalized Bayesian Probability Values for Expert Systems", *Pro. Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, (1983), pp. 198-292.

[4] Cheeseman, P. "In Defense of Probability", *Pro. Ninth International Joint Conference on Artificial Intelligence*, Los Angeles (1985), pp. 1002-1009.

[5] Genesereth, M.R. and Nilsson, N.J. *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann, (1987).

[6] Ginsberg, M.L. "Does Probability Have a Place in Non-monotonic Reasoning?", *Proc. IJCAI (1985)*, pp. 107-110.

[7] Grosof, B.N. "Non-monotonicity in Probabilistic Reasoning", *Uncertainty in A.I. Workshop Proceedings*, Philadelphia, (1986), pp. 91-98.

[8] Hanks, S. and McDermott, D. "Default Reasoning, Non-monotonic Logics and the Frame Problem", *AAAI Proceedings* (1986), pp. 328-333.

[9] Kadie, C.M. "Rational Non-monotonic Reasoning", *Proc. IJCAI*, Minn. (1988), pp. 197-204.

[10] Kane, T. "Maximum Entropy in Nilsson's Probabilistic Logic", by T. Kane, *Proceedings of IJCAI*, Detroit, (1989), pp. 452-453.

[11] McLeish, M. "Dealing with Uncertainty in Non-monotonic Reasoning", 1986, *Proceedings of the Conference on Intelligent Systems and Machines*, April (1986), Oakland University, Michigan, pp. 1-5.

[12] McLeish, M. "Probabilistic Logic: Some Comments and Possible use for Non-monotonic Reasoning", *Uncertainty in A.I. II, Machine and Pattern Recognition Series*, (1988), pp. 55-63.

[13] McLeish, M. "Further Work on Nonmonotonicity in the Framework of Probabilistic Logic with an Extension to Dempster-Shafer Theory", *Machine Intelligence and Pattern Recognition, Vol. 8*, (1989), pp. 23-34.

[14] McLeish, M. "A Note on Probabilistic Logic", *Proceedings of AAAI*, Minnesota, (1988), pp. 215-219.

[15] Nilsson, N. "Probabilistic Logic", SRI Technical Report #321 (1984) and *Artificial Intelligence, Vol. 28*, Feb. (1986), pp. 71-87.

[16] Pearl, J. *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann Publishers, (1988).

[17] Reiter, R. "A Logic for Default Reasoning", *Artificial Intelligence (13)*, (1980), pp. 81-132.

[18] Yager, R.R. "Using Approximate Reasoning to Represent Default Knowledge", *A.I. Journal 31*, (1987), pp. 99-112.

[19] Zadeh, L.A. "Fuzzy Logic and Approximate Reasoning", *Synthese 30*: (1975), pp. 407-428.

The Probability of Causal Explanation

Dekang Lin and **Randy Goebel**
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2H1
{lindek,goebel}@cs.UAlberta.ca

Abstract

We present a probabilistic theory of causal explanations, which integrates probabilistic and causal knowledge. Unlike most other approaches where a causal explanation is a hypothesis that one or more causative events occurred, we define an explanation of a set of observations to be the occurrence of a chain of causation events. These causation events constitute a scenario where all the observations are true. The underlying causal model enables us to compute probabilities of the scenarios from the conditional probabilities of the causation events.

The notion of causation event, which was first introduced in [Peng and Reggia, 1987] and was claimed to be "the crucial innovation," was nonetheless underspecified. We provide a more adequate definition here and explain its relationship to co-occurrence.

Although probabilistic causal inference is NP-hard in general, our algorithm exploits characteristics of admissible input to achieve efficient computation.

1 Introduction

Causal models have been used as a pivotal mechanism in many diagnostic reasoning systems. Diagnosis can be viewed as a process of finding the causal explanation of the observed faults. Since there are typically multiple explanations for a given set of observations, mechanisms must be provided to rank the causal explanations. The approaches to ranking explanations can be roughly categorized as follows:

Pure Bayesian Calculus: Bayes theorem provides a good basis for a explanation ranking scheme. However, as has been repeatedly pointed out in in the literature [Charniak and McDermott, 1985], the implementations of pure Bayesian calculus suffers from voracious demand for data and combinatorial explosion of explanations to be scored before reaching a conclusion.

Heuristic Scoring: Early expert systems such as CASNET [Weiss *et al.*, 1978] associate weights to causal links and uses heuristics to rank the explanations. Although these systems have achieved significant results, they have often been criticized for their use of poorly defined and unjustified weighting schemes and scoring heuristics as well as unreasonable assumptions about probability distributions.

Occam's Razor: A heuristic principle for ranking explanations is Occam's Razor, which states that everything else being equal, simpler explanations are preferred. While precise specifications of "simpler" are not always themselves simple, one straightforward definition is this: an explanation is considered simpler than another if the causative events in the former is a subset of those in the latter. This approach is taken by [Reggia *et al.*, 1983; Reiter, 1987; Allemang *et al.*, 1987]. This interpretation of simplicity is intuitively appealing, but it may not always lead to correct results [Lin and Goebel, 1990].

Causal Bayesian Calculus: Recently, there is a trend to combine causal knowledge with probabilistic knowledge by means of causal networks [Cooper, 1984; Pearl, 1988]. Instead of making independence assumptions about all the events, the known probabilistic dependencies are explicitly represented by the structure of the network.

In this paper, we present a probabilistic theory of causal explanations, which can also be classified as a causal Bayesian approach. The domain knowledge is represented by a causal network whose nodes denote events such as the presence of symptoms, disorders, or pathological states. Edges represent the potential causation events. Unlike most other approaches where a causal explanation is a hypothesis that one or more causative events occurred, we define an explanation of a set of observations to be the occurrence of a chain of causation events. These causation events constitute a scenario where all the observations are true. We show that the probabilities of the scenarios can be computed from the conditional probabilities of the causation events.

The remainder of the paper is organized as follows: in Section 2, we explain the notion of causation event. Section 3 introduces the concept of scenario. In Section 4, we show that the probability of a scenario can be computed from the conditional probability of causation events. In Section 5, we relate the problem of finding the most probable explanation to that of finding the minimal weight tree that connects all the observed observations. The latter problem is known as the Steiner Problem in Graphs [Dreyfus and Wagner, 1972]. An algorithm for solving the Steiner Problem in Graphs is briefly mentioned. The relationships between our work and other causal Bayesian approaches are discussed in Section 6. Our contributions to causal reasoning are summarized in Section 7.

2 Causation Events

The concept of causation event was first introduced by Peng and Reggia [Peng and Reggia, 1987] to explicitly represent the statement "x actually caused y." A causation event $e:c$ is true "iff both [the cause event] c and [the effect event] e occur and e is actually being caused by c [Peng and Reggia, 1987, p149]." One of the motivations for causation events being a distinct type of basic event in a probabilistic causal world is that $e:c$ cannot be expressed by a Boolean expression of the events c and e, because in situations where both c and e occur, $e:c$ may still be false. The diagram in Figure 1 illustrates the relationship between c, e and $e:c$.

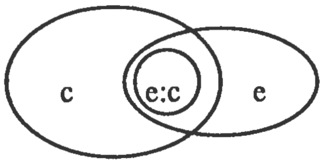


Figure 1: Relationships between the cause, effect and causation event

Unfortunately, Peng and Reggia's definition of causation event does not provide a way to judge whether a causation event occurred, because, unlike other basic events, causation events are usually unobservable. We can only observe the co-occurrence of the cause and effect.

Here we present a definition of direct causation event following a suggestion from Pearl[1]. Causation can be modeled at different levels of details [Patil, 1987]. A causation event $e:c$ is said to be direct if there are no intervening events at the current level of representation. A direct causation event may correspond to a chain of causation events at a more refined level of representation. If $e:c$ corresponds to a chain of micro causation events

$$e:i_1:\ldots:i_n:c$$

then when $e:c$ occurs, not only must c and e must occur, but the intermediate micro events $i_1, \ldots, i_n$ must also occur.

We therefore define direct causation event as follows:

Definition 2.1 (direct causation event) *A direct causation event, denoted by $c \leadsto e$, is false iff there do not exist micro events $i_1, \ldots, i_n$ at a more refined level of representation such that $c \leadsto i_1$, $i_1 \leadsto i_2, \ldots i_n \leadsto e$ is a chain of micro causation events and $e, i_1, \ldots, i_n, c$ co-occur.*

[1]Personal communication

With this definition, c and e must be true if $c \leadsto e$ is true.

In Section 4.2 we derive an upper bound for the difference between the conditional probability of the effect event $P(e|c)$ and the probability of the causation event $P(c \leadsto e|c)$, thus identifying the conditions under which $P(c \leadsto e|c)$ can be approximated by $P(e|c)$.

3 Scenario and Explanation

Definition 3.1 (causal network) *A causal network is a 3-tuple: $<E, DC, \top>$ where*

- *E a finite, non-empty set of events, each element in E is represented by a node in the network.*
- *$DC \subseteq E \times E$ is the set of potential* direct *causation events:*
  $$DC = \{x \leadsto y | x, y \in E\}.$$
  The direct causation event $x \leadsto y$ is represented by an edge from node x to node y.
- *$\top$ is a distinguished element of E which is alway true. $\top \leadsto y \in DC$ if y does not have a cause, i.e. $\nexists x \in E$, such that, $x \neq \top$ and $x \leadsto y \in DC$.*

The purpose of $\top$ is to accommodate the cases where there are multiple independent causes. $\top \leadsto x$ is true whenever x is true.

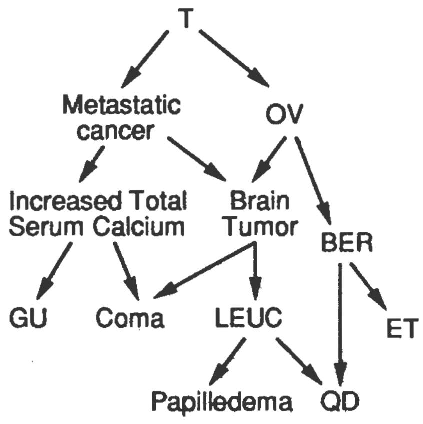


Figure 2: A causal network

Example 3.2 *Figure 2 is an example of causal network $<E, DC, \top>$, where*

$$\begin{aligned} E &= \{\top, \textit{Metastatic Cancer}, OV, \\ &\quad \textit{Brain Tumor}, \ldots, QD\}; \\ DC &= \{\top \leadsto OV, \top \leadsto \textit{Metastatic Cancer}, \\ &\quad OV \leadsto \textit{Brain Tumor}, \ldots, BER \leadsto QD\}; \end{aligned}$$

The causal explanations for the observations are scenarios where the observed events are true. Such a scenario can be regarded as a tentative reconstruction of the causal evolution which has led to the observations. Before defining the scenarios, we first introduce the following sets :

Let α be a set of causation events: $\alpha = \{x_i \leadsto y_i | i = 1, \ldots, n\}$. We define:

- $antecedents(\alpha) \overset{\text{def}}{=} \{x | x = x_i, i = 1, ..., n\}$.
- $consequents(\alpha) \overset{\text{def}}{=} \{y | y = y_i, i = 1, ..., n\}$.
- $participants(\alpha) \overset{\text{def}}{=}$
  $antecedents(\alpha) \cup consequents(\alpha)$;
- $cause\_of(\alpha) \overset{\text{def}}{=}$
  $antecedents(\alpha) - consequents(\alpha)$;

Definition 3.3 (scenario) *A scenario is a set of direct causation events and is defined recursively as follows:*

a) any direct causation event $x \rightsquigarrow y \in DC$ is a scenario.

b) if α is a scenario, $x \rightsquigarrow y \in DC$, $x \in participants(\alpha)$, and $y \notin participants(\alpha)$, then $\alpha \cup \{x \rightsquigarrow y\}$ is a scenario.

c) All scenarios are obtained from either a) or b).

Intuitively, scenarios are chains of causation events. Each scenario α is a sub-tree of the causal network. The root of the sub-tree is $cause\_of(\alpha)$. There is a directed path from the root to every other node in the tree. When there are multiple independent causes, $\top$ is the root of the tree. For example, the edges in the shaded area in Figure 3 constitute a scenario. Explanations for a set of observations are the scenarios which contain the observations.

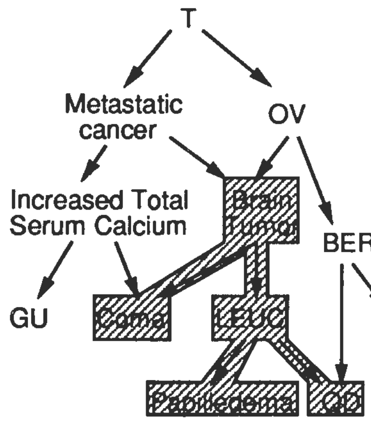


Figure 3: A scenario

Definition 3.4 (explanation) *An explanation for the set of observations O is a scenario α such that $O \subseteq consequents(\alpha)$.*

Figure 4 shows two explanations for the observation {Coma, Papilledema, QD}.

4 Probabilities of Scenarios

This section is concerned with computing the probabilities of the scenarios. Let x be an event, $x \rightsquigarrow y$ be a causation event and $\alpha = \{x_i \rightsquigarrow y_i | i = 1, \ldots, n\}$ be a set of causation events, we write $P(x)$, $P(x \rightsquigarrow y)$, and $P(\alpha)$ to denote their probabilities respectively. $P(\alpha)$ can alternatively be written as $P(x_1 \rightsquigarrow y_1, x_2 \rightsquigarrow y_2, \ldots, x_n \rightsquigarrow y_n)$. Since $\top$ is always true, we have $P(\top) = 1$. Since $\top \rightsquigarrow y$ is equivalent to y, $P(\top \rightsquigarrow y | \top) = P(\top \rightsquigarrow y) = P(y)$;

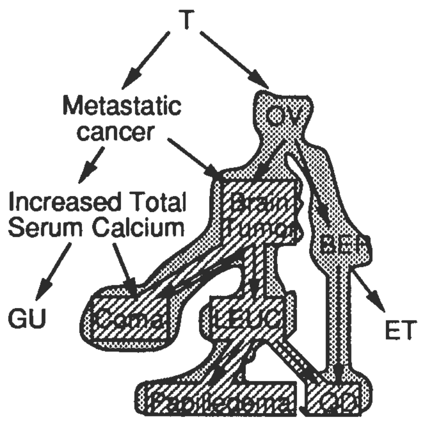


Figure 4: Two explanations

4.1 Independence Assumptions

Most practical probabilistic diagnosis systems make the following simplifying assumptions about the real world in order to apply the probability theory [Charniak and McDermott, 1985]:

1. Symptoms are independent of each other: $P(s_i, s_j) = P(s_i)P(s_j)$.
2. Symptoms are conditionally independent of each other given any disorder: $P(s_i, s_j | d) = P(s_i | d)P(s_j | d)$.

Unfortunately, these assumptions are usually false in real applications [Fryback, 1978]. It has been argued that the inadequacy of traditional Bayesian classification theories is largely due to its inability to express the causal relationships among the symptoms and disorders [Cooper, 1988]. In contrast, our explicit representation of causal structure enables us to make more realistic assumptions:

Causation Independence: This assumption states that given the cause event c, the direct causation event $c \rightsquigarrow e$ is conditionally independent of whatever caused c and other effects of c.

More formally, let $c \rightsquigarrow e \in DC$ be a direct causation event, and β be a scenario such that:

a) $c \in participants(\beta)$

b) $e \notin participants(\beta)$.

Then we assume $c \rightsquigarrow e$ and β are conditionally independent given c:

$$P(c \rightsquigarrow e | c, \beta) = P(c \rightsquigarrow e | c). \quad (1)$$

This assumption is similar to the axioms of Belief Networks in [Pearl, 1988; Poole and Neufeld, 1988].

Non-causation independence: This assumption states that causal influence is the only influence between a cause c and its effece e. I.e., the influence of c on e is insulated once $c \rightsquigarrow e$ is known to be false:

$$P(c, e | \overline{c \rightsquigarrow e}) = P(c | \overline{c \rightsquigarrow e}) \times P(e | \overline{c \rightsquigarrow e}). \quad (2)$$

4.2 Causation vs. Co-occurrence

The probability of co-occurence is always no less than the probability of causation. The following theorem gives an upper bound of the difference between conditional probabilities of co-occurrence and causation.

Theorem 4.1 *Let c and e be two nodes in the causal network and $c \leadsto e$ be a direct causal event. Then*

$$P(e|c) - P(c \leadsto e|c) < P(e)$$

Proof:

$$\begin{aligned}
& P(e|c) - P(c \leadsto e|c) \\
&= \tfrac{P(e,c,\overline{c \leadsto e})}{P(c)} \\
&= \tfrac{P(e,c,\overline{c \leadsto e})}{P(\overline{c \leadsto e})} \times \tfrac{P(\overline{c \leadsto e})}{P(c)} \\
&= P(e, c|\overline{c \leadsto e}) \times \tfrac{P(\overline{c \leadsto e})}{P(c)} \\
&= P(e|\overline{c \leadsto e}) \times P(c|\overline{c \leadsto e}) \times \tfrac{P(\overline{c \leadsto e})}{P(c)} \\
&\qquad \text{(by independence assumption (2))} \\
&= P(e|\overline{c \leadsto e}) \times P(\overline{c \leadsto e}|c) \\
&< P(e|\overline{c \leadsto e}) \\
&= \tfrac{P(e,\overline{c \leadsto e})}{P(\overline{c \leadsto e})} \\
&< \tfrac{P(e,\overline{c \leadsto e})+P(c \leadsto e)}{P(\overline{c \leadsto e})+P(c \leadsto e)} \\
&= \tfrac{P(e)}{1} \\
&= P(e) \qquad \blacksquare
\end{aligned}$$

Therefore, given that $P(e)$ is significantly smaller than $P(e|c)$, which is usually the case, $P(c \leadsto e|c)$ can be approximated by $P(e|c)$.

4.3 Probabilities of Scenarios

Given the context-freeness of causation events, the probabilities of scenarios can be obtained through the following theorem:

Theorem 4.2 *Let $\alpha = \{x_i \leadsto y_i | i = 1, \ldots, n\}$ be a scenario. Then*

$$P(\alpha) = P(cause\_of(\alpha)) \times \prod_{i=1}^{n} P(x_i \leadsto y_i | x_i)$$

Proof: This theorem is proved by induction on the structure of scenarios.
Base Case: Let $\alpha = x \leadsto y$. Then

$$\begin{aligned}
P(\alpha) &= P(x \leadsto y) \\
&= P(x \leadsto y, x) \\
&= P(x \leadsto y|x) \times P(x) \\
&= P(x \leadsto y|x) \times P(cause\_of(\alpha)).
\end{aligned}$$

Induction Step: Let $\alpha = \beta \cup \{x \leadsto y\}$ where $y \notin consequents(\beta)$, $x \in participants(\beta)$. We assume the theorem is true for β. Then,

$$\begin{aligned}
P(\alpha) &= P(\beta, x \leadsto y) \\
&= P(x \leadsto y|\beta) \times P(\beta) \\
&= P(x \leadsto y|x, \beta) \times P(\beta) \\
&\qquad (\because\ x \in participants(\beta)) \\
&= P(x \leadsto y|x) \times P(\beta) \\
&\qquad \text{(by independence assumption (1))} \\
&= P(x \leadsto y|x) \times P(cause\_of(\beta)) \\
&\qquad \times \textstyle\prod_{x' \leadsto y' \in \beta} P(x' \leadsto y'|x') \\
&\qquad \text{(by the induction assumption)} \\
&= P(cause\_of(\alpha)) \times \textstyle\prod_{x' \leadsto y' \in \alpha} P(x' \leadsto y'|x') \\
&\qquad (\because\ cause\_of(\alpha) = cause\_of(\beta)) \qquad \blacksquare
\end{aligned}$$

Corollary 4.3 *Let $\alpha = \{x_i \leadsto y_i | i = 1, \ldots, n\}$ be a scenario. Then*

$$\begin{aligned}
\log(P(\alpha)^{-1}) = {} & \log(P(cause\_of(\alpha))^{-1}) \\
& + \textstyle\sum_{i=1}^{n} \log(P(x_i \leadsto y_i|x_i)^{-1})
\end{aligned}$$

We now quantify the causal link from node x to node y by a weight $log(P(x \leadsto y|x)^{-1})$ and associate a weight $log(P(x)^{-1})$ with each node x in the causal network. $Log(P(\alpha)^{-1})$ then becomes the total weight of all the links in α plus the weight of the root.

Since maximizing $P(\alpha)$ is equivalent to minimizing $log(P(\alpha)^{-1})$, we have the following theorem:

Theorem 4.4 *The most probable explanation for a set of observations O is a sub-tree T of the weighted causal network $< E,\ DC,\ \top >$ such that:*

a) T connects all the nodes in O.

b) The total weight of the links in T plus the weight of the root of T is minimal.

5 Finding the most Probable Explanation

Following from Theorem 4.4 in the last section, the problem of finding the most probable causal explanation is a variation of a graph-theoretic problem known as the Steiner Problem in Graphs, which can be formally stated as follows:

Definition 5.1 (Steiner Problem in Graphs)
Let $G =< N, E >$ be a weighted graph, where N is the set of nodes and E is the set of edges. Each edge $e \in E$ is associated with a non-negative weight $w(e)$. Given a set of nodes $S \subseteq N$, the Steiner Problem in Graphs is to find a sub-tree $T \subseteq E$, such that,

a) all nodes in S are connected together by T;

b) $\sum_{e \in T} w(e)$ is minimal.

The minimal tree is called the Steiner Tree (connecting S).

It is well known that the Steiner Problem in Graphs is NP-Complete [Garey and Johnson, 1979, p.208]. This, however, does not imply that our model is computationally intractable. The admissible inputs must also be taken into consideration [Levesque, 1989]. A crucial observation here is that the number of observations to be explained in a single case is usually small, much smaller than the number of nodes in the network. Dreyfus and Wagner [Dreyfus and Wagner, 1972] presented an algorithm to solve the Steiner Problem in undirected Graphs with complexity $O(3^k n + 2^k n^2)$, where n is the number of nodes in the network, and k is the number of nodes to be connected.

Our algorithm is a variation of [Dreyfus and Wagner, 1972], which solves the Steiner Problem in directed graphs. The algorithm simulates a set of processes distributed over the nodes of the graph. Each process performs the same local algorithm, which consists of receiving, processing and sending messages that are transmitted across the edges. The average complexity is further reduced in our algorithm by exploiting the locality of the nodes to be connected. The nodes that are unrelated to the observations will not be involved in the computation at all. The worst case complexity is also $O(3^k n + 2^k n^2)$. Details of the algorithm can be found in [Lin and Goebel, 1990].

6 Relationship to Other Causal Bayesian Approaches

In this section, we compare our approach with three other causal Baysian approaches.

6.1 Generalized Set Covering

In the Generalized Set Covering (GSC) model [Peng and Reggia, 1987], causal knowledge is represented by a bipartite graph where links represent causal associations between disorders and manifestations. An explanation for a set of manifestations M^+ is a set of disorders D_I that are able to cause the manifestations. The explanation also implicitly assumes that disorders not in D_I are absent. Peng and Reggia present a procedure for computing $P(D_I|M^+)$ from the conditional probabilities of causation events $P(m_j : d_i|d_i)$ where m_j is a manifestation and d_i is a disorder. They then argue that the most probable D_I is *likely* to be a irredundant one, i.e. a D_I for which any proper subset of D_I is unable to cause all the manifestations in M^+. The notion of causation event, which is claimed to be "the crucial innovation [Peng and Reggia, 1987, p159]" is nonetheless underspecified. We have given a more adequate definition here and showed that $P(c \rightsquigarrow e|c)$ may be approximated by $P(e|c)$.

The main limitation of GSC is that the representation is not expressive enough to capture chaining in reasoning (e.g. "a causes b," and "b causes c," so "a causes c" indirectly where b is an intermediate event). This is too prohibitive for most real applications. The limitation to a two layer representation also results in unreasonable probability independence assumptions. First, causal relationships exist not only between disorders and symptoms, but also sometimes between a disorder and another disorder. These two disorders will then generally be dependent of each other, therefore violating the disorder independence assumption in [Peng and Reggia, 1987]. Second, when a disorder d_1 cause two symptoms s_1, s_2 via a common intermediate state, GSC will wrongly assumes that the causation events $s_1 : d_1$ and $s_2 : d_1$ are conditionally independent given d_1. For example, in Figure 5, GSC will incorrectly assume that *appendicitis causing anorexia* is conditionally independent of *appendicitis causing nausea* given *appendicitis* is known.

6.2 Belief Network

Belief networks [Pearl, 1988] are directed acyclic graphs in which each node represents a random variable, or uncertain quantity, which can take on two or more possible values. Causal explanations, which are instantiations of the variables of the causal network, are obtained by a distributed message propagation. The propagation algorithm, however, is designed for singly connected networks, (i.e. networks with no undirected loops). Although he also proposed two extensions to multiply connected networks (conditioning and clustering), both

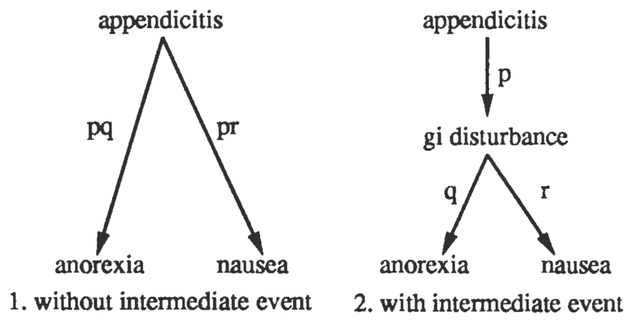


Figure 5: Indirect Causation Events may not be Conditionally Independent

methods are liable to exponential complexity [Henrion, 1987].

Another problem with the belief revision procedure is that an explanation consists of instantiations for all the variables in the network. This implies that every piece of evidence must be propagated to the entire network, even to the totally irrelevant sections of the knowledge base [Pearl, 1988, p.259]. As has been shown in [Lin and Goebel, 1990], our message passing algorithm is able to exploit the locality of the observations to be explained. The nodes that are unrelated to the observations are not activated during the message passing process.

6.3 NESTOR

NESTOR [Cooper, 1984] combines causal and statistical knowledge. Any observations believed to be independent are unaffected. Observations with known causal or associational links are modeled by causal trees that represent interactions between these dependent variables. NESTOR is unique in its use of bounded probabilities as opposed to point probabilities. Bounded probabilities do not force the expert to commit to a single value with little confidence. However, NESTOR cannot scale up because of the complexity of its scoring algorithm. Scoring a single hypothesis in NESTOR takes $O(n2^{n-k})$ time (n is the number nodes in the causal network, k is the number of observations to be explained) [Cooper, 1988] and there are potentially exponentially many hypotheses to be scored before reaching a conclusion.

In both Pearl and Cooper's network, nodes are variables, which are more general than the events represented in our theory.

7 Conclusion

We note the following as major limitations of our model:

- Explanations have to be consistent with facts and observations. Our model, however, does not provide any mechanism for consistency checking.
- We have implicitly made an assumption that the multicausal interactions are noisy OR-gates [Pearl, 1988, p.181], which cannot deal with complications resulted from multiple interacting disorders.

- The nodes in our model are propositions. It is not clear what we must do when variables or predicates have to be represented.

We summarize our contributions to causal reasoning as follows:

- The combination of causal and probabilistic knowledge defeats the straightforward application of Bayes' theorem. The probabilities we use are statistically meaningful and could potentially be determined from real-world data.
- We have given a more adequate definition of causation event and identified its relationship with cooccurrence.
- Causal chaining is captured more naturally by the connection in our approach than the covering in GSC.
- Explanations are defined to be scenarios rather than sets of causative events. This results in a more principled treatment of multiple simultaneous disorders.
- Although probabilistic inference using belief networks is NP-hard in general [Cooper, 1988], our algorithm is polynomial to the number of nodes in the networks and is exponential only to the number of observations to be explained, which, in any single case, is usually small. Moreover, the algorithm lends itself to distributed parallel processing.

Acknowledgements

We are grateful to Judea Pearl, who took the time to point out the weaknesses in an earlier paper and suggest the approach of characterizing causal events in terms of lower level micro events. This research was supported by Natural Sciences and Engineering Research Council of Canada grants G1587, INF36861, and OGP9443. Dekang Lin is supported by the Research Award from the Department of Computing Science, University of Alberta.

References

[Allemang *et al.*, 1987] Dean Allemang, Michael Tanner, Tom Bylander, and John Josephson. Computational complexity of hypothesis assembly. In *Proceedings of IJCAI87*, pages 1112–1117, 1987.

[Charniak and McDermott, 1985] Eugene Charniak and Drew McDermott. *Introction to Artificial Intelligence.* Addison-Wesley Publishing Company, 1985.

[Cooper, 1984] G. Cooper. *NESTOR: A computer-based medical diagnostic aid that integrates causal and probabilistic knowledge.* PhD thesis, Dept. Computer Scinece Standford University,, November 1984.

[Cooper, 1988] Gregory F. Cooper. Computer-based medical diagnosis using belief networks and bounded probabilities. In Perry L. Miller, editor, *Selected Topics in Medical Artificial Intelligence*, pages 85–98. Springer-Verlag, 1988.

[Dreyfus and Wagner, 1972] S. F. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.

[Fryback, 1978] D. G. Fryback. Bayes' theorem and conditional nonindependence of data in medical diagnosis. *Computers and Biomedical Research*, 11:423, 1978.

[Garey and Johnson, 1979] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* W. H. Freeman and Company, San Francisco, 1979.

[Henrion, 1987] Max Henrion. Uncertainty in Artificial Intelligence: Is probability epistemologically and heuristically adequate? In J. L. Mumpower, L. D. Phillips, Ortwin Renn, and V. R. R. Uppuluri, editors, *Expret judgement and Expert Systems*, 1987.

[Levesque, 1989] Hector J. Levesque. Logic and the complexity of reasoning. Technical Reports on Knowledge Representation and Reasoning KRR-TR-89-2, Department of Computer Science University of Toronto, 1989.

[Lin and Goebel, 1990] Dekang Lin and Randy Goebel. A minimal connection model of abductive diagnostic reasoning. In *Proceedings of 1990 IEEE Conference on Artificial Intelligence Applications*, pages 16–22, March 1990.

[Patil, 1987] Ramesh S. Patil. A case study on evolution of system building expertise: medical diagnosis. In W. Eric L. Grimson and Ramesh S. Patil, editors, *AI in the 1980s and beyond: an MIT survey*, pages 75–108. MIT Press, 1987.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann publishers, Inc., San Mateo, California, 1988.

[Peng and Reggia, 1987] Yun Peng and James A. Reggia. A probabilistic causal model for diagnostic problem solving — Part I: integrating symbolic causal inference with numeric probabilistic inference. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-17(2):146–162, 1987.

[Poole and Neufeld, 1988] David Poole and Eric Neufeld. Some probabilistic inference in prolog: An executable specification of influence. In *Proceedings of The International Symposium on Artificial Intelligence*, pages 37–54, Monterrey, N.L., Mexico, October 1988. Instituto Tecnologico y de Estudios Superiores de Monterrey.

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

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

[Weiss *et al.*, 1978] Sholom M. Weiss, Casimir A. Kulikowski, Saul Amarel, and Aran Safir. A model-based method for computer-aided medical decision making. *Artificial Intelligence*, 11:145–172, 1978.

Sequential Updating Conditional Probability in Bayesian Networks by Posterior Probability

Yang Xiang*, Michael P. Beddoes* and David Poole†
University of British Columbia, Vancouver, B.C., Canada, V6T 1W5
* Department of Electrical Engineering, yangx@ee.ubc.ca
† Department of Computer Science, poole@cs.ubc.ca

Abstract

The Bayesian network is a powerful knowledge representation formalism; it is also capable of improving its precision through experience. Spiegelhalter *et al.* [1989] proposed a procedure for sequential updating forward conditional probabilities (FCP) in Bayesian networks of diameter 1 with a single parent node. The procedure assumes certainty for each diagnosis which is not practical for many applications. In this paper we present a new algorithm (ALPP) that allows refinement of FCPs based on expert estimates of posterior probability. ALPP applies to any DAG of diameter 1. Fast convergence is achieved. Simulation results compare ALPP with Spiegelhalter's method.

1 Introduction

Much recent research is dedicated to Bayesian belief networks as an inference formalism building expert systems [Pearl 88] [Lauritzen and Spiegelhalter 88] [Heckerman *et al.* 89] [Andersen *et al.* 89]. A Bayesian network is a pair (D, P). D is a directed acyclic graph (DAG) in which the nodes represent generally uncertain variables, and the arcs signify the existence of direct causal influences between the linked variables. P is a probability distribution which quantifies the strengths of these causal influences. P is distributively stored in the network, in the form of FCPs [Pearl 88].

A knowledge based system QUALICON is currently under development, based on Bayesian networks, which can be used in assisting an E.M.G. technician in test quality control during conduction velocity studies [Xiang *et al.* 90]. The system takes qualitative features of recorded compound muscle action potentials as evidences and tries to diagnose the problems in electrode set-up.

Building the system involves two parts. The first is generating the DAG D. This task is easy and natural for the experienced medical staff. The second phase, namely the elicitation of many FCPs, they found much more difficult (and the results were quite imprecise) because, among other causes, the task seems artificial to them. They claim that it would be easier for them to supply a posterior probability (PP) distribution for the possible hypotheses in particular cases. What they give in that case would be more precise since the task is closer to their daily practice. A methodology allowing the system to improve itself through expert's PPs is badly needed.

Spiegelhalter *et al.* [1989] present a procedure for sequential updating conditional probabilities in Bayesian networks decomposed into DAGs of diameter one with a single parent node. The procedure consists of two stages. In the first stage, the FCPs for each link are elicited from the expert. The expert is asked to estimate these probabilities in form of intervals to express the imprecision of the estimation. Then each interval probability is interpreted as an imaginary sample ratio: $p(symptomA|diseaseB) = a/b$, where b is an imaginary patient population with disease B and among these patients a of them show symptom A. In the second stage, *updating* stage, whenever a new patient with disease B comes in, the corresponding sample size b is increased by 1, and the sample size a is increased either by 1 or 0 depending on if the patient shows symptom A. This *updating* stage is the main concern of this paper.

A major problem of this updating approach is the underlying assumption that when updating the link FCP $p(symptomA|diseaseB)$, the system user knows for sure whether the disease B is true or false (thus we will call the procedure $\{0,1\}$ *distribution learning*). The assumption is not realistic in many applications. A doctor would not always be 100% confident about a diagnosis he made of a patient.

In this paper, a Algorithm of Learning by Posterior Probability (ALPP) is presented which is more general than the $\{0,1\}$ distribution learning. ALPP applies to any DAG with diameter one. The DAG can itself be the whole network or be a subnet of a more sophisticated network as long as the following *DAG-completeness* condition holds: it contains all the incoming links to its child nodes as in the original net. ALPP does not assume 100% accurate posterior judgment. Instead, it utilizes the PPs of each fresh case supplied by the expert to update the FCPs of the network. We show the algorithm converges to the expert's behavior under ideal condition. When ALPP does not converge to the human consultant's posterior judgments, it is an indication of either inadequate network structure or inadequate PPs. The algorithm converges quicker than the $\{0,1\}$ distribution learning equipped with *100% accurate* posterior judgment.

The philosophy which guided this work is described in section 2. Section 3 presents ALPP and section 4 proves its convergence. The performance of the ALPP is demonstrated by simulations given in section 5.

2 Learning from Posterior Distributions

The spirit of $\{0,1\}$ distribution learning is to improve the precision of probability elicited from the human expert by learning from available data. Now the question is what do we really have in medical practice in addition to patients' symptoms? It may be possible, in some medical domain, that diagnoses can be confirmed with certainty. But this is not commonplace. A successful treatment is not always an indication of correct diagnosis. A disease can be cured by a patient's internal immunity or by a drug with wide disease spectrum. One subtlety of medical diagnosis comes from the unconfirmability for each individual patient case.

For most medical domains, the available data beside patients' symptoms are physician's subjective PPs of possible diseases. They are not distributions with values from $\{0,1\}$, but rather distributions from $[0,1]$[1]. The diagnoses appearing in patients' files are typically not the diagnoses that have been concluded definitely; they are only the top ranking diseases with physician's subjective PP omitted. The assumption of $\{0,1\}$ posterior disease distribution may, naively, be interpreted as an approximation to $[0,1]$ distribution with 1 substituting top ranking PP, and 0 substituting the rest. This approximation loses useful information. Thus a way of learning directly from $[0,1]$ posterior distribution seems more natural and anticipates better performance.

In dealing with *learning* problem in a Bayesian network setting, three "agents" are concerned: the real world (D_r, P_r), the human expert (D_e, P_e), and our artificial system (D_s, P_s). It is assumed that all 3 can be modeled by Bayesian networks. As the building of an expert system involves specifying both the topology of D and probability distribution P, the improvement can also be separated into the two aspects. For the purpose of this paper, D_r, D_e, and D_s are assumed identical, leaving to be improved only the accuracy of quantitative assignment of P_s.

An expert system based on Bayesian networks usually directs its arcs from disease (hypothesis) nodes to symptom (evidence) nodes, encoding quantitative knowledge with priors of diseases and FCPs of symptoms given diseases [Shachter and Heckerman 87, Henrion 88]. These probabilities are usually elicited from human experts.

A question which arises is whether PP is any better in quality compared to priors and FCPs also supplied by the human expert. In our cooperation with medical staff, it is found that the causal network is a natural model to view the domain, however, the task of estimating FCPs is more artificial than natural to them. Forming posterior

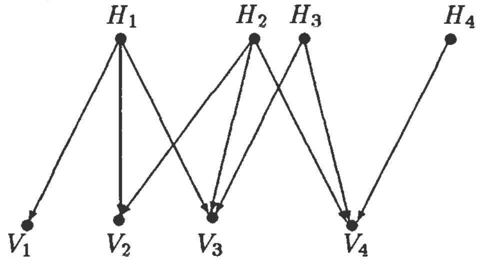


Figure 1: An example of D(1)

judgments is their daily practice. An expert is an expert in that he/she is skilled at making diagnosis (posterior judgement), not necessarily skilled at estimating FCPs. It is the expert's posterior judgment that is the behavior we want our expert system to simulate[2].

If we believe that the human expert carries a mental Bayesian network and PPs are produced by the network, it is postulated that the FCPs the expert articulates, which consists of P_s of our system, could be a distorted version of those in P_e. Also, P_e may differ from P_r in general. Thus, 4 categories of probabilities are distinguished: P_r, P_e, P_s, and the PPs produced by P_e (written as p_e). Our access to only P_s and $p_e(hypotheses|evidence)$ is assumed. We want to use the latter to improve P_s such that the system's behavior will approach that of expert.

How can PP be utilized in our updating? The basic idea is: instead of updating imaginary sample sizes by 1 or 0, increase them by the measure of certainty of the corresponding diseases. The expert's PP is just such a measure. Formal treatment is given in the following section.

3 The Algorithm for Learning by Posterior Probability (ALPP)

The following notation is used:

$D(1)$ DAGs of diameter 1 (The *diameter* is the length of the longest directed path in the DAG. An example of D(1) is given in Figure 1.);

$(D(1), P)$ Bayesian net with diameter 1 and underlying distribution P;

$H_i \in \{h_{i1}, \ldots, h_{in_i}\}$ the ith parent variable in $D(1)$ with possible values h_{i1} through h_{in_i};

$V_j \in \{v_{j1}, \ldots, v_{jm_j}\}$ the jth child variable in $D(1)$ with possible values v_{j1} through v_{jm_j};

$\mathbf{v}_{jl}$ value conjunction of all the children variables in $D(1)$ with V_j's value being v_{jl};

$b_{k_1k_2\ldots k_n}$ the imaginary sample size for joint event $h_{1k_1} \& h_{2k_2} \& \ldots \& h_{nk_n}$ being true;

$a_{l_jk_1k_2\ldots k_n}$ the imaginary sample size for joint event $v_{jl_j} \& h_{1k_1} \& \ldots \& h_{nk_n}$ being true;

[1]Note that $\{0,1\}$ denotes a set containing only elements 0 and 1, and $[0,1]$ is a domain of real numbers between 0 and 1 inclusive.

[2]We are not arguing against the usual way of encoding numerical knowledge from diseases to symptoms. The advantages of it, like simplicity in network structure, clarity of underlying causal dependency, etc. are well known.

$\delta^c_{l_j}$ impulse function which equals 1 if for the cth fresh case V_j equals v_{jl_j}, and equals 0 otherwise (superscripts denote the orders of fresh cases);

$p_r(), p_e(), p_s()$ probabilities contained or generated by $(D_r(1), P_r)$, $(D_e(1), P_e)$ and $(D_s(1), P_s)$ respectively.

A Bayesian net $(D(1), P)$ [3] is considered where the underlying distribution is composed via

$$p(h_{1k_1} \& \ldots \& h_{Nk_N} \& v_{1l_1} \& \ldots \& v_{Ml_M}) = \prod_{i=1}^{N} p(h_{ik_i}) \prod_{j=1}^{M} p(v_{jl_j} | \widehat{\mathbf{h}}_j)$$

where $\widehat{\mathbf{h}}_j$ is the conjunction of those values h_{ik_i} such that H_i is a parent variable of V_j and $h_{ik_i} \in \{h_{1k_1}, \ldots, h_{Nk_N}\}$.

Each of the FCPs is internally represented in the system as a ratio of 2 imaginary sample sizes. For child node V_1 having its parent nodes $H_1, \ldots, H_Q$ $(Q \geq 1)$, a corresponding FCP is

$$p^c_s(v_{1l_1} | h_{1k_1} \& \ldots \& h_{Qk_Q}) = a^c_{l_1 k_1 \ldots k_Q} / b^c_{k_1 \ldots k_Q}$$

where the superscript c signifies the cth updating. Only the real numbers $a^c_{l_1 k_1 \ldots k_Q}$ and $b^c_{k_1 \ldots k_Q}$ are stored. The prior probabilities for V_1's parents can be derived as

$$p^c_s(h_{ik_i}) = \frac{\sum_{k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_Q} b^c_{k_1 \ldots k_Q}}{\sum_{k_1, \ldots, k_Q} b^c_{k_1 \ldots k_Q}}$$

For a $(D(1), P)$ with M children and with all variables binary, the number of numbers to be stored in this way is upper bounded by

$$B = 2 \sum_{i=1}^{M} 2^{\beta_i}$$

where β_i is the number of *incoming* arcs to child node i. Storage saving can be achieved when different child nodes share a common set of parents.

Updating P is done one child node at a time through updating as and bs associated with the node as illustrated above. Once the as and bs are updated, the updated FCPs and priors can be derived. The order in which child nodes are selected for updating is irrelevant.

Without losing generality, we describe the updating with respect to above mentioned child node V_1. For the cth fresh case where $\mathbf{v}^c$ is the symptoms observed, the expert provides the PP distribution $p_e(h_{1k_1} \& \ldots \& h_{Nk_N} | \mathbf{v}^c)$. This is transformed into

$$p_e(h_{1k_1} \& \ldots \& h_{Qk_Q} | \mathbf{v}^c) = \sum_{h_{Q+1}, \ldots, h_N} p_e(h_{1k_1} \& \ldots \& h_{Nk_N} | \mathbf{v}^c)$$

The sample sizes are updated by

$$a^c_{l_1 k_1 \ldots k_Q} = a^{c-1}_{l_1 k_1 \ldots k_Q} + \delta^c_{l_1} p_e(h_{1k_1} \& \ldots \& h_{Qk_Q} | \mathbf{v}^c)$$

$$b^c_{k_1 \ldots k_Q} = b^{c-1}_{k_1 \ldots k_Q} + p_e(h_{1k_1} \& \ldots \& h_{Qk_Q} | \mathbf{v}^c).$$

[3]Whether it is a subnet or a net by itself is irrelevant.

4 Convergence of the algorithm

An expert is called *perfect* if $(D_e(1), P_e)$ is identical to $(D_r(1), P_r)$.

Theorem 1 *Let a Bayesian network $(D_s(1), P_s)$ be supported by a perfect expert equipped with $(D_e(1), P_e)$. No matter what initial state P_s is in, it will converge to P_e by ALPP.*

The proof is given in figure 2.

A *perfect* expert is never available. We need to know the behavior of ALPP when supported by an imperfect expert. This leads to the following theorem.

Theorem 2 *Let p^c_s be any resultant probability in $(D_s(1), P_s)$ after c updating by ALPP. p^c_s converges to a continuous function of P_e.*[4]

Proof:

(1) Continuity of priors.

Following the proof of theorem 1, the prior $p^c_s(h_{ik_i})$ converges to

$$f = \sum_{k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_Q} \sum_t p_e(h_{1k_1} \& \ldots \& h_{Qk_Q} | \mathbf{v}(t)) \frac{u(t)}{c}$$

where $p_e(h_{1k_1} \& \ldots \& h_{Qk_Q} | \mathbf{v}(t))$ is an elementary function of P_e, and so does f. Therefore, $p^c_s(h_{ik_i})$ converges to a continuous function of P_e.

(2) Continuity of FCP.

From theorem 1, $p^c_s(v_{1l_1} | h_{1k_1} \& \ldots \& h_{Qk_Q})$ converges to

$$f = \frac{\sum_t p_e(h_{1k_1} \& \ldots \& h_{Qk_Q} | \mathbf{v}_{1l_1}(t)) \frac{u_{1l_1}(t)}{c}}{\sum_z p_e(h_{1k_1} \& \ldots \& h_{Qk_Q} | \mathbf{v}(z)) \frac{u(z)}{c}}$$

where $p_e(h_{1k_1} \& \ldots \& h_{Qk_Q} | \mathbf{v}(z))$ is an elementary function of P_e.

□

Theorem 2, together with Theorem 1, says that when the discrepancy between P_e and P_r is small, the discrepancy between P_s and P_r (P_e as well) will be small after enough learning trials. The specific form of the discrepancy is left open.

The absolute value of PPs is not really important in many applications but the posterior ordering of diseases be. A set of PPs defines such a posterior ordering. We say a 100% *behavior match* between (D, P_1) and (D, P_2) if for any possible set of symptoms the two give the same ordering. The minimum difference between successive PPs of (D, P_1) defines a threshold. Unless the maximum difference between corresponding PPs from 2 (D, P)s exceeds the threshold, 100% behavior match is guaranteed. Thus as long as the discrepancy between P_e and P_r is within some $(D_r(1), P_r)$ dependent threshold, a 100% match between the behavior of P_s and that of P_e is anticipated.

[4]By 'X is a function of P_e', we mean that X takes probability variables in P_e as its independent variables which in turn themselves have [0,1] as their domain.

Without losing generality, consider the updating with respect to V_1 described in last section.
(1) Priors. Let $\{\mathbf{v}(1), \mathbf{v}(2), \ldots\}$ be the set of all possible conjuncts of evidence. Let $u(t)$ be the number of times at which event $\mathbf{v}(t)$ is true in c cases; and $\sum_t u(t) = c$. From the prior updating formula of ALPP,

$$\begin{aligned}
\lim_{c\to\infty} p_s^c(h_{ik_i}) &= \lim_{c\to\infty} \left(\frac{\sum_{k_1,\ldots,k_{i-1},k_{i+1},\ldots,k_Q} b^0_{k_1\ldots k_Q} + \sum_{k_1,\ldots,k_{i-1},k_{i+1},\ldots,k_Q} \sum_{x=1}^{c} p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}|\mathbf{v}^x)}{c + \sum_{k_1,\ldots,k_Q} b^0_{k_1\ldots k_Q})} \right) \\
&= \lim_{c\to\infty} \frac{1}{c} \left(\sum_{k_1,\ldots,j_{k-1},j_{k+1},\ldots,k_Q} \sum_t p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}|\mathbf{v}(t))u(t) \right) \\
&= \sum_{k_1,\ldots,k_{i-1},k_{i+1},\ldots,k_Q} \sum_t p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}|\mathbf{v}(t))p_r(\mathbf{v}(t)) \\
&= \sum_{k_1,\ldots,k_{i-1},k_{i+1},\ldots,k_Q} \sum_t p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}|\mathbf{v}(t))p_e(\mathbf{v}(t)) \qquad \text{(perfect expert)} \\
&= \sum_{k_1,\ldots,k_{i-1},k_{i+1},\ldots,k_Q} p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}) = p_e(h_{ik_i})
\end{aligned}$$

(2) FCPs. Let $u_{1l_1}(t)$ be the number of times at which event $\mathbf{v}_{1l_1}(t)$ is true in c cases. Following ALPP, we have

$$\begin{aligned}
\lim_{c\to\infty} p_s^c(v_{1l_1}|h_{1k_1}\&\ldots\&h_{Qk_Q}) &= \lim_{c\to\infty} \frac{a^0_{l_1k_1\ldots k_Q} + \sum_{x=1}^{c} \delta^x_{l_1} p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}|\mathbf{v}^x)}{b^0_{k_1\ldots k_Q} + \sum_{y=1}^{c} p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}|\mathbf{v}^y)} \\
&= \lim_{c\to\infty} \frac{\frac{1}{c}\sum_{x=1}^{c} \delta^x_{l_1} p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}|\mathbf{v}^x)}{\frac{1}{c}\sum_{y=1}^{c} p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}|\mathbf{v}^y)} \\
&= \frac{\lim_{c\to\infty} \frac{1}{c}\sum_t p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}|\mathbf{v}_{1l_1}(t))u_{1l_1}(t)}{\lim_{c\to\infty} \frac{1}{c}\sum_z p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}|\mathbf{v}(z))u(z)} \\
&= \frac{\sum_t p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}|\mathbf{v}_{1l_1}(t))p_r(\mathbf{v}_{1l_1}(t))}{\sum_z p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}|\mathbf{v}(z))p_r(\mathbf{v}(z))} \\
&= \frac{\sum_t p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}|\mathbf{v}_{1l_1}(t))p_e(\mathbf{v}_{1l_1}(t))}{\sum_z p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}|\mathbf{v}(z))p_e(\mathbf{v}(z))} \qquad \text{(perfect expert)} \\
&= \frac{p_e(h_{1k_1}\&\ldots\&h_{Qk_Q}\&v_{1l_1})}{p_e(h_{1k_1}\&\ldots\&h_{Qk_Q})} = p_e(v_{1l_1}|h_{1k_1}\&\ldots\&h_{Qk_Q})
\end{aligned}$$

Figure 2: Proof of Theorem 1

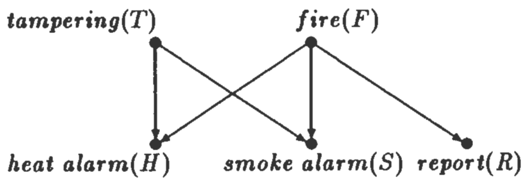


Figure 3: Fire alarm example

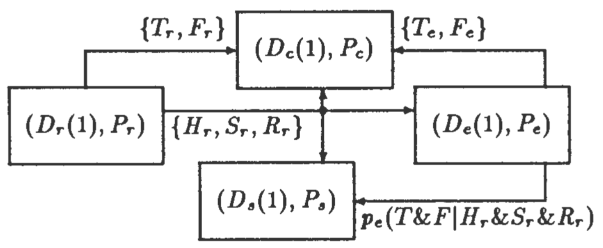


Figure 4: Simulation set-up

5 Simulation results

Several simulations were run using the example in Figure 3. It is a revised version of the smoke-alarm example in [Poole and Neufeld 88]. Here *heat alarm*, *smoke alarm* and *report* are used as evidences for estimating the likelihood of *tampering* and *fire*. Each variable, denoted by uppercase letters, takes binary values. For example, F has value f or $\overline{f}$ which signify the event *fire* being *true* or *false*.

The simulation set-up is illustrated in Figure 4. Logical sampling [Henrion 88] was used in the real world model $(D_r(1), P_r)$ to generate scenarios $\{T_r, F_r, H_r, S_r, R_r\}$. The observed evidences $\{H_r, S_r, R_r\}$ were feed into $(D_e(1), P_e)$. Posterior distributions $p_e(T\&F|H_r\&S_r\&R_r)$ was made by the expert model and were forwarded to update system model $(D_s(1), P_s)$.

To compare the performance between ALPP and $\{0, 1\}$ distribution learning, a Control model $(D_c(1), P_c)$

was constructed in the set-up. It had the same DAG structure and initial probability distribution as $(D_s(1), P_s)$ but was updated by $\{0,1\}$ distribution learning.[5] Two different sets of diagnoses were utilized in different simulation runs by $(D_c(1), P_c)$ for the purpose of comparison. In simulation 1, 2 and 3 to be described below, the top diagnosis $\{T_e, F_e\}$ made by $(D_e(1), P_e)$ was used. In simulation 4, the scenario $\{T_r, F_r\}$ was used. The former simulated the situation where posterior judgments could not be fully justified. The latter simulated the case where such justification was indeed available.

For all the simulations let P_r be the following distribution

$p(h\|f\&t)$	0.50	$p(s\|f\&t)$	0.60
$p(h\|f\&\bar{t})$	0.90	$p(s\|f\&\bar{t})$	0.92
$p(h\|\bar{f}\&t)$	0.85	$p(s\|\bar{f}\&t)$	0.75
$p(h\|\bar{f}\&\bar{t})$	0.11	$p(s\|\bar{f}\&\bar{t})$	0.09
$p(r\|f)$	0.70	$p(f)$	0.25
$p(r\|\bar{f})$	0.06	$p(t)$	0.20

and let P_s and P_c be an identical distribution with maximal error relative to P_r being 0.3. The initial imaginary sample size for each joint event $F\&T$ is set to 1. Such setting is mainly for the purpose of demonstrating the convergence of ALPP under poor initial condition. The distribution error should generally be smaller and initial sample sizes be much larger in case of practical application where the convergence will be a slowly evolving process.

	$(D_e(1), P_e)$	$(D_s(1), P_s)$		$(D_c(1), P_c)$	
trial No.	diag. rate	behv. mat. rate	max. err. S-E	behv. mat. rate	max. err. C-E
0			0.30		0.30
1~25	68%	60%	0.14	48%	0.21
26~50	76%	96%	0.10	12%	0.25
51~100	80%	100%	0.06	36%	0.27
101~200	76%	100%	0.03	33%	0.28

Table 1: Simulation 1 summary

Simulation 1 was run with P_e being the same as P_r which assumed a perfect expert. The results are depicted in Table 1. The diagnostic rate of $(D_e(1), P_e)$ is defined as A/N where N is the base number of trials and A is the number of trials where the top diagnosis agrees with $\{T_r, F_r\}$ simulated by $(D_r(1), P_r)$. The behavior matching rate of $(D_s(1), P_s)$ relative to $(D_e(1), P_e)$ is defined as B/N where B is the number of trials in which $(D_s(1), P_s)$'s diagnoses give the same ordering as $(D_e(1), P_e)$'s do. The behavior matching rate of $(D_c(1), P_c)$ to $(D_e(1), P_e)$ is similarly defined.

The results show convergence of probability values in P_s to those in P_e (maximum error(S-E) $\rightarrow$ 0). The behavior matching rate of $(D_s(1), P_s)$ increases along with the convergence of probabilities and finally $(D_s(1), P_s)$ achieved exactly the same behavior as that of $(D_e(1), P_e)$. An interesting phenomenon is that, despite $P_e = P_r$, the diagnostic rate of $(D_e(1), P_e)$ was only 76% in the total 200 trials. Though the rate is dependent of the particular (D, P), it is expected to be less than 100% in general. In terms of medical diagnosis, this is because some disease may manifest through unlikely symptoms, making other diseases more likely. In an uncertain world with limited evidence, mistakes in diagnoses are unavoidable. More importantly, P_s converged to P_e under the guidance of this 76% correct diagnoses while P_c did not. The maximum error of P_c remained about the same throughout the 200 trials and the behavior matching rate of $(D_c(1), P_c)$ was low. Similar performance of $(D_c(1), P_c)$ was seen in the next 2 simulations. This shows that under the circumstances where good experts are available but confirmations to diagnoses are not available, ALPP is robust while {0,1} distribution learning will be misled by the errors in diagnoses. This is not surprising since the assumption underlying {0,1} distribution learning is violated. We will gain more insight into this from the results of simulation 4 below.

An imperfect expert was assumed in simulation 2 (Table 2). The distribution P_e differed from P_r up to 0.05. Because of this error, P_s converges to neither P_e (as shown in Table 2) nor P_r. But the error between P_s and P_e approached a small value (about 0.07) such that after 200 trials the behavior of P_s matched that of P_e perfectly.

	$(D_e(1), P_e)$	$(D_s(1), P_s)$		$(D_c(1), P_c)$	
trial No.	diag. rate	behv. mat. rate	max. err. S-E	behv. mat. rate	max. err. C-E
0			.300		.300
1~100	84%	82%	.058	32%	.272
101~200	86%	92%	.122	43%	.287
201~300	80%	100%	.067	32%	.290
301~400	83%	100%	.076	36%	.292

Table 2: Simulation 2 summary

If the discrepancy between P_s and P_r is further increased so that the threshold discussed in last section is crossed, $(D_s(1), P_s)$ will no longer converge to $(D_e(1), P_e)$. This is the case in simulation 3 (Table 3) where the maximum error and root mean square error (rms) between P_e and P_r were 0.15 and 0.098 respectively. The rms error was calculated over all the priors and conditional probabilities of P_e and P_r. We introduced rms error for interpretation of simulation 3 because maximum error itself, when not approaching to 0, did not give good indication of the distance between the two.

The simulation shows that the behavior matching rate of P_s and P_e is quite low (43% after 475 trials). Since the diagnostic rate of P_e is also lower (77%), one could ask which one is better. One way of viewing this is to compare the diagnostic rates. It is observed that, among Ps, P_c and P_e, no one is superior than others if *only* top diagnosis is concerned. More careful examination can be

[5]Here we have extended $\{0,1\}$ distribution learning to D(1).

$(D_e(1), P_e)$		$(D_s(1), P_s)$				
trial No.	diag. rate	diag. rate	behv. mat. rate	rms err. S-E	rms err. S-R	max. err. S-E
0				.170	.169	.39
1~25	80%	84%	20%	.086	.079	.17
26~75	74%	74%	40%	.071	.068	.11
76~175	73%	73%	53%	.050	.083	.087
176~375	79%	79%	46%	.059	.072	.095
376~475	78%	78%	43%	.061	.071	.119
$(D_e(1), P_e)$		$(D_c(1), P_c)$				
trial No.	diag. rate	diag. rate	behv. mat. rate	rms err. C-E	rms err. C-R	max. err. C-E
0				.170	.169	.39
1~25	80%	80%	32%	.110	.091	.20
26~75	74%	74%	38%	.110	.092	.16
76~175	73%	73%	38%	.098	.092	.15
176~375	79%	79%	23%	.100	.090	.15
376~475	78%	78%	26%	.096	.084	.15

Table 3: Simulation 3 summary

obtained by comparison of distances among models. It turns out that the distance (S-E) and distance (S-R) are smaller than the distance (E-R) with corresponding rms errors 0.061, 0.071 and 0.098 respectively.

The above 3 simulation assumed that only the subjective posterior judgments were available. In simulation 4, it was assumed that the correct diagnosis was also accessible. This time, $(D_c(1), P_c)$ was supplied with the scenario generated by $(D_r(1), P_r)$. P_e was the same as P_r.

The results (Table 4) showed that ALPP converged much quicker than {0,1} distribution learning even the latter had access to "true" answers to the diagnostic problem. After 1100 trials, $(D_s(1), P_s)$ reduced its maximum error from $(D_e(1), P_e)$ to 0.041 and matched the latter's behavior perfectly, while $(D_c(1), P_c)$ was still on its way of convergence with its error about 2 times larger and its behavior matching rate 80%.

$(D_e(1), P_e)$		$(D_s(1), P_s)$		$(D_c(1), P_c)$	
trial No.	diag. rate	behv. mat. rate	max. err. S-E	behv. mat. rate	max. err. C-E
0			.300		.300
1~100	88%	95%	.130	60%	.375
101~600	78%	98%	.048	72%	.045
601~1100	78%	93%	.052	61%	.075
1101~1500	79%	100%	.041	80%	.079
1501~1700	81%	100%	.025	85%	.093

Table 4: Simulation 4 summary

Real world scenarios could be distinguished as being *common* or *exceptional*. An expert with knowledge about the real world tends to catch the common and to ignore the exceptional. Thus the diagnostic rate will never be 100%. This is the best one could do given the limited evidence. The PPs provided by the expert contain the information about the entire domain, while a scenario contains only the information about this particular scene. Thus, although both $(D_s(1), P_s)$ and $(D_c(1), P_c)$ converged, the former converged quicker. This difference in convergence speed is expected to emerge wherever the diagnosis is difficult and the diagnostic rate of the expert is low.

6 Remarks

An algorithm of learning by PP distribution (ALPP) for sequential updating probability in Bayesian networks is presented. ALPP is based on any DAGs of diameter 1. After a network is constructed through elicitation of expert knowledge (qualitatively the dependency in the domain and quantitatively the FCPs), ALPP can be applied to improve it towards the expert's behavior. Several features of ALPP can be appreciated through the theoretical analysis and simulation results given in the paper.

- ALPP does not assume 100% posterior knowledge about the "true" answer to a diagnostic problem as does the {0,1} distribution learning [Spiegelhalter *et al.* 89]. When only expert's posterior judgments are available, ALPP converges to expert's behavior while {0,1} distribution learning will be misled by unavoidable error made in expert's diagnoses due to the violation of its underlying assumption.
- When both expert's posterior judgments and "true" answers are accessible, ALPP converges faster than {0,1} distribution learning due to the richer information contained in expert's posterior judgments.
- ALPP is tolerant to human consultants who are good but imperfect. When ALPP can not converge after many learning trials, it is an indication of inadequate DAG structure or inadequate posterior judgments.
- Computation of ALPP is simple.
- ALPP offers the possibility of combining the expertise from multiple experts, although this requires further research.

Acknowledgements

This work is supported by Operating Grant 583200 and OPPO044121 from NSERC. Thanks are also directed to A. Eisen and M. MacNeil for valuable discussions.

References

[Andersen *et al.* 89] S. K. Andersen, K. G. Olesen, F. V. Jensen and F. Jensen, "HUGIN - a shell for building Bayesian belief universes for expert systems," *Proceedings, IJCAI - 89*, 1080-1085, 1989.

[Heckerman *et al.* 89] D. E. Heckerman, E. J. Horvitz and B. N. Nathwani, "Update on the Pathfinder project," *13th Symposium on computer applications in medical care*, 1989.

[Henrion 88] M. Henrion, "Propagating uncertainty in Bayesian networks by probabilistic logic sampling", J. F. Lemmer and L. N. Kanal (Edt), *Uncertainty in Artificial Intelligence 2*, Elsevier Science Publishers, 1988.

[Pearl 88] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann.

[Lauritzen and Spiegelhalter 88] S. L. Lauritzen and D. J. Spiegelhalter, "Local computation with probabilities on graphical structures, and their application to expert systems," *J. Roy. Stat. Soc.*, B, 50, 157-244, 1988.

[Poole and Neufeld 88] D. Poole and E. Neufeld, "Sound probabilistic inference in Prolog: an executable specification of influence diagrams," *I SIMPOSIUM INTERNACIONAL DE INTELIGENCIA ARTIFICIAL*, Oct. 1988.

[Shachter and Heckerman 87] R. D. Shachter and D. E. Heckerman, "Thinking backward for knowledge acquisition", *AI Magazine*, 8:55-62, 1987.

[Spiegelhalter *et al.* 89] D. J. Spiegelhalter, R. C. G. Franklin, and K. Bull, "Assessment, criticism and improvement of imprecise subjective probabilities for a medical expert system," Proceedings, Fifth workshop on uncertainty in artificial intelligence, 335-342, Aug. 1989.

[Xiang *et al.* 90] Y. Xiang, A. Eisen, M. MacNeil and M. P. Beddoes, "QUALICON: Artificial intelligence in quality control and diagnosis of neuromuscular disease", to appear in American Academy of Neurology Annual Meeting, May, 1990.

On the Representation of Concurrent Actions in the Situation Calculus

Jay C. Weber
Lockheed AI Center (O/96-20, B/259)
3251 Hanover Street
Palo Alto, California 94304-1191
jay@icsi.berkeley.edu

Abstract

There is no doubt that the most influential representation of action in artificial intelligence has been the situation calculus. Nevertheless, in recent years it has had many detractors who argue that the situation calculus is epistemologically inadequate to represent many action scenarios, in particular those with multiple agents performing concurrent actions. The main result of this paper is that the situation calculus *can* represent many of these scenarios, and it is in fact as epistemologically adequate as a discrete temporal logic with instantaneous actions. However, the point of this paper is not to promote the situation calculus, but to demonstrate how intuitive arguments about representational power can be subtly wrong, obscuring more important considerations of simplicity and understandability.

1 Introduction

The *situation calculus* [McCarthy, 1968] was the first general, formal representation of action and effects. Its fundamental ontology of situations and operators formed the basis for seminal work on planning [Fikes and Nilsson, 1971; Sacerdoti, 1977; Fikes *et al.*, 1972], and it continues to influence formal and algorithmic theories of action reasoning [Lifschitz, 1987; Wilkins, 1982]. Yet we have seen many arguments against this representation. The most common and significant objection has been that "...it is impossible to represent concurrent actions in [the] situation calculus" [Pelavin and Allen, 1986]. Other researchers are equally blunt: "...in the situation calculus, concurrent actions aren't allowed" [Morgenstern and Stein, 1988]. The argument is also explicated in the following quote:

> Most early work in action planning assumed the presence of a single agent acting in a static world. In the formulation of these problems, the world was considered to be in one of a potentially infinite number of states and actions were viewed as mappings between these states. However, the formalisms developed did not allow for simultaneous action, and as such are inadequate for dealing effectively with most real-world problems that involve other agents and dynamically changing environments [Georgeff, 1987].

This paper takes a conciliatory view of the relationship between the representation of concurrent, non-interfering actions and the ontology of the situation calculus. The principal result herein is that despite statements in the above quotes, the situation calculus ontology is as capable of representing concurrent actions as discrete temporal calculi, which are often used because of their ability to represent concurrent actions. This comparison takes the form of a straightforward syntax translation function on sentences in the two approaches. The apparent lack of support for concurrent actions in the situation calculus stems from how the formalism is generally used, where action constants completely determine resulting situations.

2 The Situation Calculus

The situation calculus is based on three ontological concepts: the *situation*, the *propositional fluent*, and the *situational fluent*. A situation is an instantaneous state or "snapshot" of the domain being modeled. A propositional fluent is a set of situations, i.e. the strongest property that is true of the members of that set. A situational fluent is a mapping from situations to situations, i.e. how the application of an action transforms each situation into a new situation.

These concepts can be represented as first-order logic objects, predicates, and functions, respectively, as is done in dynamic logic; however, fluents are usually *reified* by representing them as objects and relating them to situations by a special **holds** predicate and a special **result** function. This paper uses the notation presented by Lifschitz [Lifschitz, 1987], which is as follows:

- $\mathtt{holds}(p, s)$ denotes whether the propositional fluent denoted by p contains (is true of) the situation denoted by s.
- $\mathtt{result}(a, s)$ denotes the resulting situation from applying the situational fluent denoted by a to the situation denoted by s.

Arguments against this representation, as mentioned in the introduction, revolve around the fact that only a single situational fluent is allowed as an argument to the result function. Since a given situational fluent completely determines the situation mapping, there is no room for the influences of additional actions. Nevertheless, the next section shows how a straightforward use of the situation calculus ontology *can* represent the class of concurrent actions with cumulative effects, i.e. *non-interfering* [Georgeff, 1987] actions.

3 Causal Rules and Action Types

Traditionally, a causal rule in the situation calculus has the following form:[1]

$$\mathtt{holds}(\mathtt{p_1}, s) \land \ldots \land \mathtt{holds}(\mathtt{p_n}, s) \\ \rightarrow \mathtt{holds}(\mathtt{e}, \mathtt{result}(\mathtt{a}, s)),$$

where $\mathtt{p_1}, \ldots, \mathtt{p_n}$ are sufficient preconditions for the performance of action a to imply the effect e in the resulting situation. Representations based on this form of causal rule, of which there have been many [Hayes, 1971; Fikes and Nilsson, 1971; Lifschitz, 1987, ...], are subject to the criticisms of the previous section. The situational fluent on the right-hand side of this rule is called an "action", performed by a single force in an otherwise static world. Concurrent actions are indeed incompatible with this interpretation of action.

However, a slightly different version of this causal rule does allow concurrent actions. First, replace the situational fluent *constant* on the right-hand side with a constrained variable, in the following way:

$$\mathtt{holds}(\mathtt{p_1}, s) \land \ldots \land \mathtt{holds}(\mathtt{p_n}, s) \land \mathtt{type}(a, \mathtt{A}) \\ \rightarrow \mathtt{holds}(\mathtt{e}, \mathtt{result}(a, s)).$$

This rule says that when the preconditions $\mathtt{p_1}, \ldots, \mathtt{p_n}$ hold and a situational fluent of type A is applied to a situation, the effect e will hold in the result. An example rule of this form would be:

$$\mathtt{holds}(\mathtt{loaded}(g), s) \land \mathtt{type}(a, \mathtt{trigger}(g)) \\ \rightarrow \mathtt{holds}(\mathtt{not}(\mathtt{loaded}(g)), \mathtt{result}(a, s)).$$

This rule asserts that when a loaded gun's trigger is pulled, it will become unloaded. However, it does not make assertions about other dynamic properties; the rule may be applied without completely determining the resulting situation. This is because the situational fluent on the right-hand side is not completely determined.

If the situational fluent argument to the result function is called an *action*, as it typically is when using the situation calculus, then the second argument to the type relation would be an *action type*. However, an action type like $\mathtt{trigger}(g)$ plays a role that is commonly called simply an action. These two notions of action present in the above formula correspond to the situation calculus and explicit temporal logic notions of action, thus illuminating the major semantic difference between the two formalisms. As the above formula shows, the explicit temporal logic notion of action is relatively easy to capture in the situation calculus; this point will be elaborated in the next section.

[1]In this and other logical sentences in this paper, italicized letters are implicitly universally quantified over the entire sentence.

Now that we can express partial resulting situations, it becomes a simple matter to combine the partial mappings of concurrent actions by asserting two situational fluent types. For example, suppose that concurrent with pulling a gun's trigger, a fragile object is dropped, even by some other agent somewhere else. If we have a rule that says that object will become broken, i.e.:

$$\mathtt{holds}(\mathtt{fragile}(o), s) \land \mathtt{type}(a, \mathtt{drop}(o)) \\ \rightarrow \mathtt{holds}(\mathtt{broken}(o), \mathtt{result}(a, s)) \quad ,$$

and we assert that some situational fluent is both the drop and pull trigger types, plus that the right preconditions hold in some situation, i.e.:

$$\mathtt{type}(\mathtt{f_5}, \mathtt{trigger}(\mathtt{g_3})) \land \mathtt{type}(\mathtt{f_5}, \mathtt{drop}(\mathtt{o_4}))$$

$$\mathtt{holds}(\mathtt{loaded}(\mathtt{g_3}), \mathtt{s_7}) \land \mathtt{holds}(\mathtt{fragile}(\mathtt{o_4}), \mathtt{s_7}),$$

then both of the above causal rules may be invoked, resulting in the following conclusions about effects:

$$\mathtt{holds}(\mathtt{broken}(\mathtt{o_4}), \mathtt{result}(\mathtt{f_5}, \mathtt{s_7}))$$

$$\mathtt{holds}(\mathtt{not}(\mathtt{loaded}(\mathtt{g_3})), \mathtt{result}(\mathtt{f_5}, \mathtt{s_7})).$$

Note that although the above example illustrates how concurrent actions may be represented, it requires that they be *non-interfering*, i.e. that their joint effects are (at least) the union of their individual effects. The difficulty with representing interfering actions is not unique to this use of the situation calculus; indeed it is a difficulty with any causal reasoning system that performs local inferences about effects, because the reasoning takes on a nonmonotonic character (although there have been knowledge-intensive monotonic solutions [Georgeff, 1987; Weber, 1989]). Remedying this, in fact, involves a solution to the infamous *qualification problem* [McCarthy, 1977]. Similarly, the qualification problem is the most important issue in the representation of *overlapping* actions, which are also not supported by the above approach.

Cooperating actions are easy to represent by stipulating multiple situational fluent types in a single causal rule. For example, if the gun is aimed at the same time as the trigger is pulled, then the target will be hit:

$$\mathtt{holds}(\mathtt{loaded}, s) \land \mathtt{type}(a, \mathtt{trigger}) \land \mathtt{type}(a, \mathtt{aim}) \\ \rightarrow \mathtt{holds}(\mathtt{target\text{-}hit}, \mathtt{result}(a, s)).$$

Therefore, the tandem effects of these concurrent actions are properly stronger than their individual effects.

A reasonable argument can be made that the above forms of causal rules are not properly part of the situation calculus, since the typical explications and applications of the situation calculus provide causal rules that do not involve action types. Nearly every example that I have seen places situational fluent *constants* in the causal rules, which completely determine the resulting situation, and therefore suffers from the standard criticisms. The point should at least be taken that the situation calculus requires far less of an overhaul than is typically called for in order to support concurrent actions and the related concepts of partial predictions and external events.

This point is also made, quite masterfully, by Schubert [Schubert, 1989]. In fact, he also presents a technique for representing and reasoning about concurrent actions in the situation calculus. His approach differs from mine in that he employs a composition function $\texttt{costart}(\mathtt{a}_1, \mathtt{a}_2)$ that produces the action object of performing action objects $\mathtt{a}_1$ and $\mathtt{a}_2$ concurrently. Such terms can be used as the action argument of the **result** function, and combined effects proven about the resulting state. The key is that these actions do not alone determine resulting states; assertions (or lack thereof) about the **result** function itself (on any action) support reasoning about external influences, implicit effects, and ambiguous outcomes. This paper's action types approach also supports this reasoning, but with the following advantages:

1. **type** assertions are simpler than **costart** definitions and terms;
2. type assertions can be given incrementally, whereas actions in **costart** terms are the *only* ones carried out by the agents of interest;[2]
3. assertions about action types are analogous to assertions about action occurrences in popular explicit temporal logics.

The analogy between action type and occurrence assertions is in fact strong enough to construct a translation, which the next section does to demonstrate the expressiveness of the situation calculus using action types.

4 Explicit Time in the Situation Calculus

One of the more popular temporal formalisms is the discrete, explicit time-line temporal calculus [Shoham, 1988; Georgeff, 1987; Morgenstern and Stein, 1988; Goodwin, 1988, ...]. This formalism starts with a partitioning of a global time-line into a countable number of what I call *moments* (avoiding a commitment to either points or non-overlapping intervals). Since moments are countable in number, moment constants are usually taken to be the integer constants, thereby inheriting integer operations like addition and a total ordering. Also like the integers, the time line is typically unbounded both from above and below.

Reified propositions are asserted to be true over moments using the relation "T" (many different relation names have been used here, including "**holds**", which I have avoided here to distinguish it from the situation calculus relation). For example, we could write that the reasoning agent is grasping a particular block at time 5 in the following way:

$$\mathtt{T}(\mathtt{grasping}(\mathtt{b}_2), 5).$$

Actions are also reified objects, and asserted to occur over moments using the relation "**occurs**", as in the following example:

$$\mathtt{occurs}(\mathtt{pickup}(\mathtt{b}_2), 4).$$

Causal rules are often addressed in a generation form, where the truth of appropriate preconditions, together with the occurrence of an action, imply that an effect is true over the succeeding moment, e.g.

$$\begin{gathered}\mathtt{T}(\mathtt{clear}(b), m) \wedge \mathtt{occurs}(\mathtt{pickup}(b), m) \\ \rightarrow \mathtt{T}(\mathtt{grasping}(b), \mathtt{succ}(m)),\end{gathered}$$

where **succ** produces the succeeding moment, usually defined to be $m + 1$. Incidently, the existence of this function and its convenience in causal rules is the reason why temporal logics are usually discrete.

In the last section, I said that explicit temporal logic actions correspond to action types in the situation calculus. Here I make that correspondence more precise by defining a translation function Φ from the above temporal logic to the situation calculus. The definition starts with a translation from moments to distinguished situations, i.e. for all moments m, $\Phi(\ulcorner m \urcorner) = \ulcorner \mathtt{s}_m \urcorner$, where a total ordering is imposed on these situations via the axiom:

$$\mathtt{s}_{\mathtt{succ}(m)} = \mathtt{result}(\mathtt{f}_m, \mathtt{s}_m).$$

Thus there is both a situation $\mathtt{s}_m$ and a situational fluent $\mathtt{f}_m$ defined for each moment m (although since they are defined in terms of each other, these extra objects exist only for notational convenience). Assertions about the truth of properties are translated to the holding of propositional fluents in the obvious way:

$$\Phi(\ulcorner \mathtt{T}(p, m) \urcorner) = \ulcorner \mathtt{holds}(p, \mathtt{s}_m) \urcorner,$$

and the occurrence of actions are translated to type assertions by:

$$\Phi(\ulcorner \mathtt{occurs}(a, m) \urcorner) = \ulcorner \mathtt{type}(\mathtt{f}_m, a) \urcorner.$$

Applying Φ to the literals of a sentence extends Φ's domain to causal rules, i.e.:

$$\begin{gathered}\Phi(\ulcorner \forall m[\mathtt{T}(p, m) \wedge \mathtt{occurs}(a, m) \rightarrow \mathtt{T}(e, \mathtt{succ}(m))] \urcorner) = \\ \ulcorner \forall m[\mathtt{holds}(p, \mathtt{s}_m) \wedge \mathtt{type}(\mathtt{f}_m, a) \\ \rightarrow \mathtt{holds}(e, \mathtt{result}(\mathtt{f}_m, \mathtt{s}_m))] \urcorner.\end{gathered}$$

The translation can be extended to theories composed of causal rules and assertions about particular scenarios. Thus the popular temporal logic given above can be expressed as a notational variant of the situation calculus. Not only does this refute the expressibility argument for discrete temporal formalisms over the situation calculus, but the translation Φ is so straightforward that it can be done in linear time and space – giving algorithms similar computational properties as well.

5 Frame Axioms for Action Types

It should be obvious that traditional frame axioms [McCarthy and Hayes, 1969] do not make sense for action types. Such explicit statements that all properties not directly affected by an action do not change could be written for the action objects that actually do the mappings between states, but since we do not make assertions about the identity of these objects (only that they

[2]The completeness of action objects can be used to make inferences about non-actions, e.g. if an agent performs a costart(walk, talk) then it doesn't perform a think. Certainly there are domains where this sort of inference is useful, such as when interpreting stories, but in general the completeness property is unnecessary or even unreasonable. Thus an approach where one must explicitly assert completeness, like when using action types, is more appropriate.

belong to types), these frame axioms have no motivation nor use.

Interestingly enough, McCarthy's approach of *minimizing abnormality* [McCarthy, 1984] applies directly to causal rules using action types. This approach adds the persistence axiom:

$$\neg\mathtt{ab}(p, a, s) \wedge \mathtt{holds}(p, s) \rightarrow \mathtt{holds}(p, \mathtt{result}(a, s)),$$

and then minimizes (applies the closed-world assumption to) the ab predicate, i.e. action objects only affect "abnormal" propositions, which are relatively rare. When we assert the type of an action object, causal rules imply that some propositions do change, overriding the default and asserting abnormality. However, this approach does not resolve ambiguities when abnormality assumptions conflict [Hanks and McDermott, 1986], which is generally considered undesirable.

Lifschitz's approach of minimizing causes [Lifschitz, 1987] can also be applied to causal rules with action types. Essentially, this approach supplies the persistence axiom (law of inertia):

$$((\mathtt{precond}(q, a) \wedge \neg\mathtt{holds}(q, s)) \vee \neg\mathtt{causes}(a, p, \mathtt{false})) \\ \wedge\mathtt{holds}(p, s) \rightarrow \mathtt{holds}(p, \mathtt{result}(a, s)),$$

and then minimizes the causes and precond predicates, i.e. if the reasoner does not know that the action causes a proposition to change, or the reasoner knows that the action has a false precondition, then the proposition will not change. This differs from the minimizing disability approach in that the extensions of the causes and precond predicates are not sensitive to the changes that are actually observed, like ab. In Lifschitz's use of the situation calculus, this means two things: 1. minimizing causes does not infer ramifications [Lifschitz, 1987; Weber, 1988], and 2. it resolves the ambiguity discovered by Hanks and McDermott. However, when used with action types, Lifschitz's approach takes on the following properties instead: 1. ramifications can be inferred, and 2. the Hanks and McDermott problem comes back! Since type assertions do not exactly determine the action objects, there exist unique action objects for every situation transition. Thus situations are implicitly part of the action objects themselves, and are therefore considered in the minimization. This produces properties more like minimizing abnormality than minimizing causes.

The question remains as to how the frame problem can be adequately addressed in a situation calculus representation using action types. Not surprisingly, given the comparisons in this paper, this question is similar to one faced by engineers of time-line representations. One answer has been to define exactly what actions could cause a proposition to become false, and then minimize action occurrences [Morgenstern and Stein, 1988; Sandewall, 1988; Weber, 1988]. In other words, one can assume that a proposition is unchanged if all actions that could cause it to change can be assumed to not occur. Essentially, this is implemented by merely circumscribing the occurs predicate in the presence of a complete causal theory.

This solution can be applied to action types in an analogous way, by providing *explanation closure* axioms [Schubert, 1989] and then minimizing membership of action objects in types. For example, suppose we know that an object must be dropped or hit to become broken, i.e.

$$\neg\mathtt{holds}(\mathtt{broken}(o), s) \wedge \mathtt{holds}(\mathtt{broken}(o), \mathtt{result}(a, s)) \\ \rightarrow \mathtt{type}(a, \mathtt{drop}(o)) \vee \mathtt{type}(a, \mathtt{hit}(o)), \tag{1}$$

and that

$$\neg\mathtt{holds}(\mathtt{broken}(\mathtt{vase1}), \mathtt{s}) \wedge \mathtt{type}(\mathtt{a5}, \mathtt{sleep}).$$

Then after circumscribing the type predicate, we can infer $\neg\mathtt{type}(\mathtt{a5}, \mathtt{drop})$, $\neg\mathtt{type}(\mathtt{a5}, \mathtt{hit})$, and when combined with the contrapositive of (1), we infer $\neg\mathtt{holds}(\mathtt{broken}(\mathtt{vase1}), \mathtt{result}(\mathtt{a5}, \mathtt{s}))$. This approach of minimizing types is interesting in how it parallels approaches to default membership in more general taxonomic reasoning.

Minimizing occurrences, whether they are expressed using type or occurs, can produce unreasonable conclusions when actions *generate* other actions [Weber, 1989]. For example, consider the generation rule:

$$\mathtt{T}(\mathtt{loaded}(g), m) \wedge \mathtt{occurs}(\mathtt{trigger}(g), m) \\ \rightarrow \mathtt{occurs}(\mathtt{fire}(g), m),$$

which says that if the trigger is pulled on a loaded gun it will fire. If it is known that the trigger is pulled yet not known whether the gun is loaded, then the circumscriptive conclusion that the gun was not fired implies that the gun must not have been loaded – pulling this fact seemingly out of thin air. The problem is the default assumptions about nonoccurrences are insensitive to other facts unless they *contradict* the assumption. In the above example, the occurrence of a trigger should block assumptions about the occurrence of a fire. Morgenstern and Stein [Morgenstern and Stein, 1988] accomplish this by excluding from the minimization any action occurrence that is implied by an antecedent with at least one true conjunct. Since this solution is syntactic, however, it is easily to construct counterexamples to its intent (insignificant or tautological conjuncts, for instance).

A more robust solution to the frame problem, and this extends beyond the situation calculus, lies in more complex models of defeasible arguments for the persistence of facts, where bodies of *evidence* are identified that justify default conclusions or numeric beliefs [Nute, 1986; Loui, 1987; Pearl, 1988; Weber, 1989].

6 Conclusions

In this paper, I have shown how to represent concurrent, non-interfering, non-overlapping actions using a simple version of the situation calculus. The conclusion to be drawn from this result is *not* that the causal reasoning field should return to the use of the situation calculus; the fact that the situation calculus is ordinarily used in a restrictive way (with situational fluent constants in result terms) is an indication that perhaps a more intuitive notational variation is in order. The Φ translation to the situation calculus presented here involves two distinct interpretations of action in one representation, which is certainly more confusing than the single interpretation in an explicit-time version.

The appropriate conclusion is that the central issue in choosing a causal representation, at least as far as concurrent actions are concerned, is not expressibility but how easily and reliably a representation can be used to express adequate axioms for the reasoning task. To that end, other representations such as frames and causal graphs may be more adequate than the sentential approaches described here.

Acknowledgments

Suggestions from Len Schubert, James Allen, and Bill Mark were greatly appreciated during the development of the ideas in this paper. Also thanks to both (anonymous) reviewers, who suggested that I discuss how frame axioms relate to action types.

References

[Fikes *et al.*, 1972] Richard E. Fikes, P. Hart, and Nils J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3(4), 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.

[Georgeff, 1987] Michael P. Georgeff. Actions, processes, and causality. In *Proceedings of the 1986 Workshop on Reasoning about Actions and Plans*, 1987.

[Goodwin, 1988] Scott D. Goodwin. Reasoning in temporal domains: dealing with independence and unexpected results. In *Proceedings of CS/CSI*, pages 46–52, June 1988.

[Hayes, 1971] Patrick Hayes. A logic of actions. In *Principles for Designing Intelligent Robots*, pages 495–519, Metamathematics Unit, University of Edinburgh, 1971.

[Hanks and McDermott, 1986] Steve Hanks and Drew McDermott. Default reasoning, nonmonotonic logics, and the frame problem. In *Proceedings of AAAI-86*, August 1986.

[Lifschitz, 1987] Vladimir Lifschitz. Formal theories of action: preliminary report. In Frank M. Brown, editor, *The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop*, Morgan Kaufman, April 1987.

[Loui, 1987] Ronald P. Loui. *Theory and Computation of Uncertain Inference and Decision.* PhD thesis, University of Rochester Computer Science Department, September 1987.

[McCarthy, 1984] John McCarthy. Applications of circumscription to formalizing common sense knowledge. In *Proceedings of the AAAI Non-Monotonic Workshop*, pages 295–324, October 1984.

[McCarthy, 1977] John McCarthy. Epistemological problems of artificial intelligence. In *Proceedings of IJCAI-77*, pages 1034–1044, Cambridge, MA, 1977.

[McCarthy, 1968] John McCarthy. Programs with common sense. In M. Minsky, editor, *Semantic Information Processing*, pages 403–417, MIT Press, 1968.

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

[Morgenstern and Stein, 1988] Leora Morgenstern and Lynn Andrea Stein. Why things go wrong: a formal theory of causal reasoning. In *Proceedings of AAAI-88*, pages 518–523, August 1988.

[Nute, 1986] Donald Nute. *LDR: A Logic for Defeasible Reasoning.* Technical Report ACMC Research Report 01–0013, Advanced Computational Methods Center, University of Georgia, Athens, Georgia, 1986.

[Pelavin and Allen, 1986] Richard Pelavin and James F. Allen. A logic for planning in temporally rich domains. *IEEE transactions*, August 1986.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufman, 1988.

[Sacerdoti, 1977] Earl D. Sacerdoti. *A Structure for Plans and Behavior.* Elsevier, New York, 1977.

[Sandewall, 1988] Erik Sandewall. Non-monotonic entailment for reasoning about time and action – part I: sequential actions. August 1988. working paper.

[Schubert, 1989] Lenhart Schubert. Solving the original frame problem without frame axioms or nonmonotonicity. Working paper, 1989.

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

[Weber, 1989] Jay C. Weber. The myth of domain-independent persistence. May 1989. Presented at the First International Workshop on Human and Machine Cognition, Topic: The Frame Problem, and submitted to AAAI-90.

[Weber, 1988] Jay C. Weber. *A Versatile Approach to Action Reasoning.* Technical Report 237, Computer Science Department, University of Rochester, March 1988.

[Wilkins, 1982] David E. Wilkins. *Domain Independent Planning: Representation and Plan Generation.* Technical Report, SRI International, August 1982.

The Net-Clause Language - A Tool for Describing Network Models

Zdravko Markov, Christo Dichev, Lydia Sinapova

Institute of Engineering Cybernetics and Robotics - BAS
1113 Sofia, "Acad.G.Bonchev" str. bl.29A, BULGARIA

Abstract

The paper presents briefly a network modeling environment, called Net-clause language. The language is designed for describing distributed computation schemes without centralized control and using unification as a basic processing paradigm. The language is capable to implement a data-driven inference, combined with a kind of default mechanism. The authors' attention is focused mainly on the logical interpretation of the data-driven inference as a resolution procedure, working on Horn clauses. The default mechanism is illustrated by examples in the framework of default reasoning.

1 Introduction

Most of the network models used in AI are just notations (e.g. semantic networks). The real working network systems are mainly connected with parallel distributed processing (PDP), well developed in the field of numeric computation. Even modern connectionism, which attempts to generalize the PDP is also based on numeric computation. Opposed to numeric computation are the symbolic approaches in AI - the methods for problem solving, including automatic deduction. Here we present an on-going research project aimed at employing network approaches in symbolic computation.

We propose a language for describing network models, based on symbolic processing. The basis of the language is the network formalism presented in [Markov,1989], where it was considered as an extension of Prolog. Its applications in the field of graphical object representation and as a connectionist modeling tool are shown there. In the present paper the formalism (here called *Net-clause language*) is further elaborated and its interpretation as a reasoning scheme is shown.

The Net-clause language is an extension of the standard Prolog. It uses the syntax of Prolog terms and its semantics is aimed at modeling graph like structures (networks), consisting of nodes and links. The nodes specify procedures unifying terms, and the links are channels along which the terms are propagated. The language is designed for *describing distributed computation schemes, without centralized control and using unification as a basic processing mechanism*.

2 Net-clause Programming

The net-clause programs describe *networks*. The basic constructors of net-clause programs are the *net-clauses*. A net-clause is a sequence of *nodes*, syntactically represented as structures (complex terms), separated by the special functor ":". The *network links* are implicitly defined by specifying variables in the nodes. Specifying one and the same variable in several nodes defines an explicit link between them. The variables in the Net-clause language are called *net-variables*. Their basic feature is that the scope of their binding and sharing is the whole net-clause. This is achieved by abandoning the mechanism of variable copying, required by the standard Prolog. Links in the network can be established also dynamically by unifying (sharing) net-variables. As the net-clauses are not copied when they are accessed (like Prolog clauses), the dynamically established links are valid globally among all net-clauses until the network is active. Dynamic links are the only way to organize the communication between different net-clauses, since the variables in each net-clause are local.

The language provide means for fixing the created by sharing net-variables dynamic links. Thus the structure of the network could be also built in a data-driven manner. Actually these means are *learning procedures*, based on generalization of ground instances. However this aspect of the language if out the scope of the present paper.

Net-clauses are constructed of three types of nodes:

1. *Free nodes*. These are structures used to access net-variables, inside and outside the net-clause.

2. *Spreading activation nodes*. They have the following syntax:

node(X1,...,Xn,M,< procedure >)

The purpose of the node procedure is to unify terms, particularly to bind variables, which in turn could propagate both data (terms) and control (activation) further among other nodes in the network. **M** is an integer number and its semantics is to define a *threshold*, determining the amount of data required to activate the procedure. **Xi** are net-variables which serve as channels for term propagating. They can be viewed as *excitatory links* and as *inhibitory links* for the activation of the procedure. The excitatory links are represented as *simple (ordinary) variables* and the inhibitory links are represented as *negated variables* (written as **~ Xi**). The procedure is activated if the difference between the number of the bound simple variables and the number of the bound negated ones is equal to **M**. When defining a spreading activation node the condition **M > 0** is required. This ensures that the procedure can not be activated "by definition", i.e. at least one variable binding is needed for that purpose. Actually binding a simple variable decrements **M**, and binding a negated one increments it, thus the procedure is activated when **M = 0**. In such a way **M** can be used to indicate dynamically the number of bindings of **Xi**.

The activation condition conforms to one general principle - it should depend only on the net-variable unification. This principle is adopted to ensure that the network realizes *distributed computation without centralized control*. This means that the unification is not only the basic data processing mechanism, but also a mechanism for specifying data-driven control in the network.

The procedures have associated truth values, i.e. then may succeed or fail. They can be specified in one of the following forms:

* **Term1 = Term2**. This is an explicit unification of net-variables. It succeeds if **Term1** is unified with **Term2**;

* **< functor > (< sequence of terms >)**. A procedure in this form succeeds if this structure is unified with an existing free node in the network.

* **External Procedure**. This is a Prolog goal, activating some Prolog predicate defined outside the network.

* **Sequence of Procedures**. This is a conjunction or disjunction of several simple procedures (each having one of the above three forms).

Procedures can be also activated outside the procedural nodes to initiate the network. Initiating the network means unifying some net-variables. This is done through the free nodes, called as Prolog goals. In the presence of proper conditions some of the spreading activation nodes activate procedures, which in turn unify more variables and activate new procedures.

3. *Default nodes*. These nodes are specified by the following syntax (the procedure is optional):

default(X,Term,< procedure >), or
default(X,Term)

The activation condition for the optionally specified procedure is connected with net-variable sharing (unifying two net variables). When an attempt to unify **X** with another variable **Y** is made the procedure is activated (if any). If the procedure terminates successfully or unconditionally (if it is not present) **Y** is unified with **Term**. If the procedure fails the variable **Y** becomes free and the original unification (the unification between **X** and **Y**) succeeds.

As it is seen by the definition a net-variable can propagate a term not being bound to it. This means that one and the same variable can be used both as a *spreading activation* channel and *activation by-need* channel. Moreover, the term being bound to a net-variable has a greater priority than the needed (shared) one, since once bound the variable can not be shared later. This is a very important feature of the language, because it ensures the implementation of a non-monotonic inference scheme which is discussed later.

To illustrate the features of the Net-clause language we shall briefly discuss a simple example (described in detail in [Markov,1989], for a different activation scheme). Let us consider the following net-clause program:

```
/* Free Nodes - Network Inputs */
edge(A,B,S1,L1):
edge(B,C,S2,L1):
edge(C,D,S1,L1):
edge(D,A,S2,L1):
edge(B,E,S2,L2):
edge(E,F,S1,L1):
edge(F,A,S2,L2):
edge(E,G,S3,L3):
edge(G,A,S4,L4):

/* General case of a four-side figure */
node(A,B,E,G,4,fig(four_side_figure)):          /* 1 */

/* Hidden node checking perpendicularity */
node(S1,S2,2,p(S1,S2,P)):                       /* 2 */

/* Non-perpendicular figures */
node(A,B,E,F,~ P,4,fig(parallelogram)):         /* 3 */
node(A,B,C,D,~ P,4,fig(rhombus)):               /* 4 */

/* Perpendicular figures */
node(A,B,E,F,P,5,fig(rectangular)):             /* 5 */
node(A,B,C,D,P,5,fig(square)):                  /* 6 */

/* Free Node - Network Output */
fig(Fig).
```

```
/* Procedure calculating perpendicularity */
p(X,Y,true):-0 is (X-Y) mod 90,!.
p(_,_,_).
```

The program describes a network for recognition of planar four-side geometric figures. The figures are represented as a collection of edges with parameters - written as free nodes. The shared variables in these nodes represent the common vertices and the geometric constraints (parallel and same-length edges). The shared variables, grouped in the spreading activation nodes, represent a "part-of" hierarchy. Thus, unifying the free nodes with the nodes of a particular instance, the bound net-variables activate the corresponding class of figures.

The example shows a way of using hidden nodes (an intermediate layer between input and output nodes) in such networks. Node 2 is activated when the net-variables **S1** and **S2** (representing the slopes of the corresponding edges) are bound. If the condition for perpendicularity is present, then the procedure "**p**" binds the net-variable **P** to "true", thus activating the "perpendicular" classes and suppressing the "non-perpendicular" ones (because of the inhibitory link ~P).

The network is activated by specifying the edges of sample figures as net-clause goals. The corresponding class is obtained by the free node "**fig**", playing the role of a network output. Some examples of the network activation are shown below.

```
?- edge(1,2,0,20),edge(2,3,45,30),edge(3,4,0,20),
  edge(4,1,45,30),fig(X).
X=parallelogram

yes
?- edge(1,2,0,20),edge(2,3,90,20),edge(3,4,0,20),
  edge(4,1,90,20),fig(square).

yes
?- edge(1,2,0,20),edge(2,3,50,20),edge(3,4,0,20),
  edge(4,1,50,20),fig(X).
X=rhombus

yes
?- edge(a,b,0,20),edge(c,d,45,30),edge(d,e,10,40),
  edge(e,a,50,60),fig(X).
X=four_side_figure
```

3 Logic and Net-clause Programming

The logic interpretation of the net-clauses is discussed in the framework of the spreading activation scheme, considered as a kind of *data-driven inference*. We define a correspondence between the *Horn clause language* [Lloyd, 1984] and the Net-clause one. A Horn clause program is translated into a network described by several net-clauses. Each program clause is translated into a net-clause, where the clause head is represented by a spreading activation node and the clause body - by a collection of free nodes. The shared variables in the clause head and body are translated into shared net-variables in the net clause. The goal clause is represented as a net-clause built out of free nodes, which can share variables, thus introducing the means to share variables in the original Horn clause goal. Finally, the unit clauses are represented as a net-clause goal, which activates the net-clause program. All different net-clauses communicate through the procedure calls to the free nodes, and the whole process is governed by the spreading activation scheme.

To illustrate the above correspondence we discuss an example of how a Horn clause program can be transformed into a net-clause one. Consider the Horn Clause program:

```
1. p(a,b) <--
2. p(c,b) <--
3. p(X,Z) <-- p(X,Y),p(Y,Z)                    (1)
4. p(X,Y) <-- p(Y,X)
5. <-- p(a,c)
```

This program is transformed to the following net-clause program (the Horn clauses and net-clauses are numbered correspondingly).

```
1,2. ?- p(a,b),p(c,b).
3.  node(X,Y,Z,3,p(X,Z)):
     p(X,Y):
     p(Y,Z).                                   (2)
4.  node(X,Y,2,p(X,Y)):
     p(Y,X).
5.  p(a,c):[].
```

The program **(1)** has clear declarative meaning, however there is no Prolog system, which is able to find a refutation for it (most of the Prolog systems will go into an infinite loop). This is because of the fixed computation and search rules used in the practical implementations of the SLD-resolution. The program **(2)** runs successfully on the net-clause interpreter. It realizes an inference process directed from the unit clauses to the goal clause.

To activate the net-clauses **(2)** we specify the goal **(1,2)**:

```
?- p(a,b),p(c,b).
```

The inference process in terms of resolution refutation initiated by the above goal is illustrated in **Fig. 1**. The proof procedure shown in the figure is based on resolution where the refutation procedure is initiated by the unit clause resolution. In fact the Net-clause goal, which represents a set of unit positive clauses is the input for the resolution process. So, the data-driven inference might be interpreted in terms of *set of support strategy* [Chang and Lee,1973; Stickel,1986]. Let us denote all net-clauses representing the program clauses (net-clauses 3 and 4 in the example) by **S**. The net-clause goal (unit positive clauses) can be viewed as a set of support **T**. Then the left net-clauses in the program comprise **S-T**. All derived clauses are ancestors of the input data and the derivation of the empty clause is a result of resolving any derived or input clause with a goal clause.

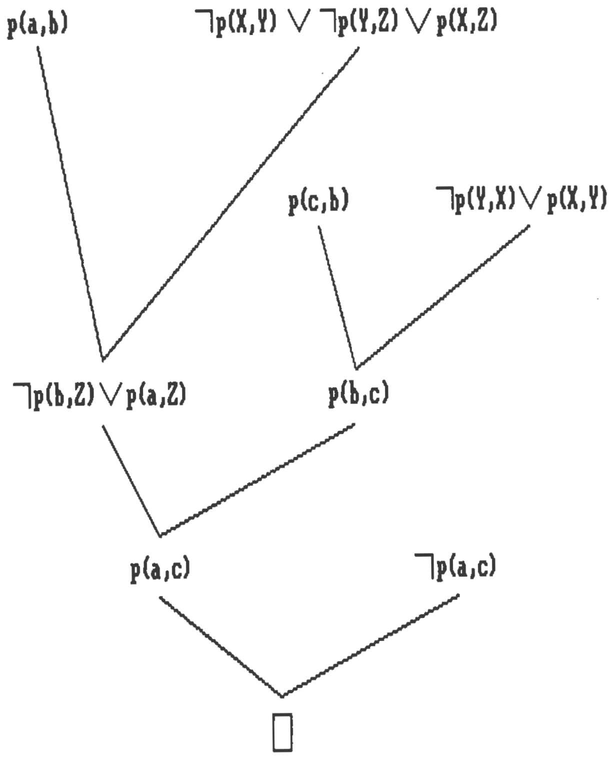


Figure 1

Using the clauses 1-4 of the program **(1)** non-ground goals could be proved too. For example we can ask with the goal

<-- p(X,Y)

and obtain all solutions. To do that we change the free node 5 with the following one:

p(X,Y): node(X,Y,2,write(p(X,Y))).

In such a way we define a node which can indicate the satisfaction of the goal, printing the answer substitutions. Hence we can obtain all possible solutions by the following question:

```
?- p(a,b),p(c,b),nl,fail.
p(a,c)
p(b,c)
p(c,b)
p(c,a)
p(b,a)
p(a,b)
```

The example considered in the present section outlines the general scheme of using the Net-clause language for data-driven inference. The program discussed is quite simple - the number of positive and negative literals are equal. Therefore the hard problems of handling the non-determinism when resolving the complementary literals are avoided. However the non-determinism is an important feature of the Horn clause programs. Therefore we try to extend the semantics of the Net-clause language in order to cover the whole class of Horn clause programs.

The underlying idea to solve the problem of non-determinism is introducing a kind of *lazy evaluation* of the net-variables. The lazy evaluation in our computational model is performed under decentralized control of the inference process. This is a way to avoid the back-tracking, which is the bottle-neck of the classical inference algorithms and an undesirable feature in the framework of the decentralized computational paradigm of the Net-clause language.

The lazy evaluation can be described in terms of *producers* and *consumers* [Reddy,1986]. In our inference scheme the producers bind net-variables and the consumers check the consistency of the these bindings. The need of lazy evaluation arise when a conjunction of two goals **Gi** and **Gj** (two free nodes) that share variables have to be evaluated. Since there is no explicitly defined producer-consumer relationship between the goals **Gi** and **Gj**, either of them can produce bindings of their shared variables. However if the bindings are inconsistent any of the goals **Gi** and **Gj** become consumers waiting for new bindings to be produced. Each shared variable is instantiated only after a consistent binding is produced. The spreading activation node in a net-clause has no access to the net-variables before they are instantiated. This ensures that inconsistent solutions in local sense (the solutions that do not satisfy all conditions of a given net-clause) will never be produced.

To implement the lazy evaluation scheme we extend the semantics of the net-clause attaching to it a kind of a *local memory*. The local memory stores solutions inferred currently by the spreading activation node. The new solutions obtained during the inference process are added to the old ones. By introducing lazy evaluation the net-clauses become more independent in the overall inference process. Thus we abandon completely the centralized control of the inference process.

The concept of the local memory is realized in the framework of the net-variables. Each net-variable is provided with a space for storing all terms which have been instantiated to it. We call this *set of instantiations* of the variable. When two variables are shared their set of instantiations are used to achieve consistency of the variable bindings. To prevent the propagation of inconsistent bindings the spreading activation node is activated only after checking the consistency of the accumulated instantiations. The consistency check of two shared variables is a procedure searching for elements in both sets of instantiations satisfying both constraints. The activation of the node is delayed until the consistency checking procedure terminates with positive result for all shared variables. When this procedure fails the net clause waits for new solutions. In addition if a net-clause does not contain any shared variables it could be activated when any of its free nodes is unified independently.

The lazy evaluation with decentralized control fits well to the data-driven inference adopted here. The producer-consumer constraints are applied and new data are generated only when "enough" input data are available. In such a way the size of the output data (locally generated solutions) is reduced to the necessary minimum.

Summarizing, the data-driven inference process is divided into two independent processes:

1. Local inference of solutions (new data) and propagating them among the net-clauses in the program. This process is governed locally by the spreading activation nodes.

2. Supplying the net-clause program with data and keeping a track of the currently inferred solutions. This process can be organized in a stepwise manner. At each step a data item is supplied to the network and the solution (if any) is checked whether it satisfies the defined goal.

4 Default Reasoning in Net-clauses

The activation-by-need mechanism exhibits some interesting properties, which can be viewed in the framework of *default reasoning*.

Default reasoning is a quite general concept in AI, which shares the features of a paradigm and an implementational principle. Generally it is based on the natural commonsense idea: *"In the absence of any information to the contrary assume that..."*. Our discussion follows the Reiter's interpretation of that principle [Reiter,1980] and further explores it in the framework of the net-clause language.

One of the defaults commonly used in knowledge representation and frame-based languages is the *default assignment to variables*. The general form of this default rule is the following (the variable types are omitted for simplicity):

$$\frac{\nvdash \mathbf{EyP(x1,\ldots,xn,y)}}{\mathbf{P(x1,\ldots,xn,<\text{default value for } y>)}} \qquad (1)$$

This rule is applied in the process of some deductive inference, when the attempt to find a value for y, satisfying the predicate P has failed. It postulates that in such cases the default value for y should be assumed.

The default node in the net-clause language can be used to define a similar rule, which is applied during the process of spreading activation. As we have seen the spreading activation control mechanism discussed in the previous section, supports a kind of data-driven inference. At each inference step a unification is performed, i.e. a procedure is activated. The procedure has the form of the predicate P in rule (1) and its arguments are net-variables. If the unification is successful some of the net-variables might be bound and some - shared. The values binding the net-variables are deduced from previous unifications (activation of previous procedures). Sharing a net variable can be seen as an unsuccessful attempt to deduce its value and that is the point to use a default value. These considerations lead to a definition of a net-clause version of rule (1):

$$\frac{\nvdash \mathbf{(bound\ Y)\ p(X1,\ldots,Xn,Y)}}{\mathbf{p(X1,\ldots,Xn,<\text{default value for } Y>)}} \qquad (2)$$

The semantic of rule (2) is that *"if we fail to bind the variable Y, then assume the default value for it"*. This rule is defined by the following net-clause (a free node and a default node):

p(X1,...,Xn,Y):default(Y,< default value for Y>).

The actual default value could be an arbitrary term.

Specific features of the described default scheme are:

1. The default value for a net-variable could be another net-variable. In such a way we define a *default link* in the network. The underlying idea is the following: *"if data can not be obtained through a particular channel, use another one"*.

2. An immediate consequence of the above feature is that the defaults can be organized in a *hierarchy*. This can be used for implementing a property-inheritance mechanism within a frame-like knowledge representation scheme.

3. The default value is attached to a net-variable, but not to a predicate, which should be deduced. This means that *the scope of the default is the whole network*.

4. The procedure in the default node allows for drawing *alternative default conclusions*.

Default reasoning is non-monotonic in nature since new evidence may invalidate previously made default conclusions. To accomplish non-monotonicity it is necessary to provide means for overriding a default assignment by currently available data. In the presented reasoning scheme non-monotonic behavior is achieved by *not binding the variables to their corresponding default values*. The default value of a variable is propagated along the network upon request, but it is not bound to the variable itself. If new data arrive these data are associated to the corresponding variables and further propagated along the network, thus overriding the assumed defaults. This mechanism allows also for distinguishing ordinary conclusions from default ones, which may be useful for estimating the plausibility of the derived conclusions.

The possibility to organize the default nodes in a hierarchy allows for drawing default conclusions on the basis of previously made default assumptions, which in our view is very natural for the common-sense reasoning. If default hierarchies are used it is possible that one and the same variable is subsequently associated to different default terms depending on the order of data input and the defined default hierarchies. The non-monotonic behavior of the default reasoning mechanism is illustrated by the example given bellow.

```
/* A net-clause with free nodes and default nodes */

a(X):b(Y):c(X,Y):
default(X,Y):
default(Y,default_value).

/* Examples of the net-clause activation. Comments are
   given in italics */

?- a(Default),c(X,Y).

Default=default_value      a simple example of deducing
X=default_value            default values
Y=default_value

yes

?- a(Default),b(hard_fact1),a(hard_fact2),c(X,Y).

Default=default_value      non-monotonic assignment:
X=hard_fact2               input data overrides the initial
Y=hard_fact1               default value

yes

?- a(Default_1),b(hard_fact),a(Default_2).

Default_1=default_value    non-monotonic assignment:
Default_2=hard_fact        "hard_fact" overrides the initial
                           default value and specifies a
                           new default value for the node
                           "a(X)"
```

The next example illustrates a way of organizing a default hierarchy:

```
a(X):b(Y):c(Z):
default(X,Y,p1(Y)):
default(Y,Z,p2(Z)):
default(Z,defZ).

p1(X):-write('(p1,Y) = '),read((t,X)).
p2(X):-write('(p2,Z) = '),read((t,X)).
```

In this program the input-output variables **X**,**Y** and **Z** can obtain their values form three sources:

(a) by binding the default value, specified in the corresponding default node;

```
?- b(val_Y),a(X).
X=val_Y
```

(b) after a successful execution of the procedure in the corresponding default node (in absence of the conditions in item a). The procedure **p1** succeeds and binds **Y** to **val_X**;

```
?- a(X).
(p1,Y) = t,val_X.
X=val_X
```

(c) using the default value of the default value. This is an example of using the default hierarchy. (Specifying "fail" on read means fail of the default procedure).

```
?- a(X).
(p1,Y) = fail.
(p2,Z) = t,val_Y.
X=val_Y

?- a(X).
(p1,Y) = fail.
(p2,Z) = fail.
X=defZ
```

The above examples show the priority of the different sources of the default values - decreasing from cases (a) to (c).

5 Conclusion

The net-clause language described in the present paper was inspired by some practical problems, such as representation of visual objects (discussed in [Markov,1989]), natural language understanding, semantic networks and modeling default reasoning. In [Markov,1989] the ideas behind the language have been discussed in the framework of connectionism.

In the present paper we have attempted to clarify the semantics of the net-clause language, discussing its relation to logic. However our goal was not to implement data-driven inference on Horn-clauses, but to create a language for network modeling. Later it appeared that the net-clauses are similar to Horn clauses and in the same time offer more complex semantics.

We have only outlined the default reasoning scheme in the net-clause language. Currently considerable efforts are directed toward the application of the language in Natural Language parsing. We see this as a promising application area, mainly because the proposed default rules are not only a theoretical framework, but a real programming environment.

Another important issue focusing currently our attention is how to program in the net-clause language. Since the language falls in the class of distributed processing ones, programming is considered mostly as learning. We see the learning procedures to be implemented at the basic processing level of the language (the unification algorithm) by using the "learning by example" paradigm (e.g. generalization of ground terms).

Despite the distributed nature of the computational process of the Net-clauses, the procedures are activated sequentially. However it is important to note that the order of their activation is not explicitly specified by control structures of the language (as in Prolog) but entirely depends on the data which are processed by the network (the result of the term unification). Actually this is one of the basic paradigms of the Net-clause language - the data-driven control.

An important aspect of PDP schemes is *parallelism*. Though we implement the Net-clause language in a purely sequential environment (extending sequential Prolog) it has the basic features to be implemented on a parallel architecture. We think that having in hand a distributed computational scheme the transition to parallel processing is only an implementational step. For this purpose it is necessary to organize the execution of the node procedures as separate parallel processes. In this scheme the activation conditions in the language can serve as synchronization conditions for the processes.

References

[Chang and Lee,1973] Chang C.-L., Lee R. C.-T. Symbolic logic and mechanical theorem proving. Academic Press, London, 1973.

[Lloyd,1984] Lloyd J. W. Foundations of Logic Programming. Springer-Verlag, 1984.

[Markov,1989] Markov Z. A framework for network modeling in Prolog, Proceedings of IJCAI'89, pp.78-83.

[Stickel,1986] Stickel M. E. An Introduction to Automated Deduction. In Bibel W., Jorrand Ph. (Eds.) Fundamentals of Artificial Intelligence. An advanced Course, Springer-Verlag, 1986.

[Reiter,1980] Reiter R, A logic for default reasoning, Artificial Intelligence, Vol.13(1980), 81-132.

[Reddy,1986] Reddy U. S. On the relationship between logic and functional languages. In Degroot D., Lindstrom G. (Eds.) Logic programming - Functions, Relations, and Equations, Prentice-Hall, 1986.

Temporal Integration

André Trudel
Jodrey School of Computer Science
Acadia University
Wolfville, Nova Scotia, Canada, B0P 1X0
Trudel@AcadiaU.CA

Abstract

A first order logic suitable for representing a world which changes over time must deal with two types of temporal information. The first is information which is true or false at a point. For example, 'the book is on the table at 3pm'. A standard method of representing such information is to associate the temporal information with the time point via a relation, say *true*. For example, we could write *true(3pm, on(book,table))*. The second type of temporal information that needs to be represented is information that is true or false over an interval. For example, 'the book was on the table between 3 and 4pm'. The usual method of representing such information is to extend the approach used for representing point based information. The interval based information is directly associated with a time interval via a relation, say *true*. For example, we could write *true(3pm,4pm, on(book,table))*.

In this paper, we present a different approach for representing information associated with an interval. We do not directly associate these assertions with intervals. The representation of these assertions is based on the assumption that what is true at every point in an interval completely determines what is true over the interval. We use the Riemann integral to relate an interval with its internal points.

1 Introduction

Every aspect of the world around us changes with time. Therefore if we are to use a computer to represent and reason about the real world, we must take time into account. There are two types of temporal information that need to be represented. The first is information which is true or false at a point. Examples are:

The book is on the table at 3pm.
John is running at time t.

A standard AI technique for representing information that is true or false at a point is to use a relation, say *true*, to associate information with a time point. For example, the above examples can be represented as:

true(3pm, on(book,table)).
true(t, running(John)).

The second type of temporal information that needs to be represented is information that is true or false over an interval. Examples are:

The book was on the table between 3 and 4 pm.
John ran without stopping between times t1 and t2.
John ran a while between times t1 and t2.

We consider the above information to be of a qualitative nature. There is also quantitative information that can be true or false over an interval. For example:

John ran a kilometer between 3 and 4 pm.
John ran without stopping at a velocity v(t) between times t1 and t2.

The traditional AI approach for representing information associated with an interval is to extend the method used for representing point based information. A relation, say *true*, is used to directly associate interval based information with a time interval. For example,

The book was on the table between 3 and 4 pm.

could be represented as

true(3pm, 4pm, on(book,table)).

Allen [1984], McDermott [1982], and Shoham [1988] use a variant of this approach in their logics.

In this paper, we present a different approach for representing information associated with an interval. We define a first order logic called GCH suitable for representing information which changes over time. GCH uses the *true* relation described above to represent point based information. GCH differs from previous temporal logics in the way it represents interval based information. We do not directly associate these assertions with intervals. The representation of these assertions is based on the assumption that what is true at every point in an interval completely determines what is true over the interval. We use the Riemann integral to relate an interval with its internal points. The Riemann integral is used for representing both quantitative and qualitative interval based information.

Below we show how GCH uses the Riemann integral to represent qualitative and quantitative information. We then outline GCH's syntax and semantics. We conclude with examples.

2 Qualitative information

Qualitative information associated with an interval is represented by integrating over the truth values at each point in the interval. We use the example "the house is red over the interval (t_1, t_2)" to illustrate the use of the Riemann integral. We treat "house is red" as a 0–1 function as follows. The function $house(red)(t)$ equals 1 at the time point t if "house is red" is true at t, and 0 otherwise. The following integral

$$\int_{t_1}^{t_2} house(red)(t)\ dt \quad = \quad t_3$$

gives the length of time (i.e., t_3) that $house(red)$ is true during the interval (t_1, t_2). The above is written as

$integral(t_1, t_2, house(red), t_3)$

in GCH. For example, "the house is red over the interval (0,10)" is written as

$integral(0, 10, house(red), 10)$.

The above specifies that the house is red at each point in the interval (0,10). It is important to note that it is not associating $house(red)$ with the interval (0,10).

We now show how the qualitative examples presented earlier are represented in GCH. The example

The book was on the table between 3 and 4 pm.

is similar to the red house example and is represented as

$integral(3, 4, on(book, table), 1)$

in GCH.

The second example

John ran without stopping between times t_1 and t_2.

is represented as

$integral(t_1, t_2, running, t_2 - t_1)$.

GCH's representation of the above example is significantly different from traditional approaches. For example, in Shoham's logic the above is written as

$TRUE(t_1, t_2, ran\text{-}without\text{-}stopping)$.

The above directly associates "ran without stopping" with the interval (t_1, t_2).

The third example

John ran a while between times t_1 and t_2.

is true if and only if the integral of running over the interval (t_1, t_2) is non-zero:

$integral(t_1, t_2, running, t_3) \quad \wedge \quad t_3 > 0$.

3 Quantitative information

We use the Riemann integral in the standard fashion to represent quantitative information associated with an interval. For example, we integrate the velocity function associated with "running" to get the displacement due to "running" over an interval.

To represent quantitative information associated with an interval, we use the *true* and *integral* relations plus the following relations:

$velocity(B, P_1, P_2)$,

$displacement(P_1, P_2, B, P_3)$,

and

$acceleration(B, P_1, P_2)$.

The relation $velocity(B, P_1, P_2)$ is true if and only if the velocity of B at time point P_1 is P_2. For example,

$velocity(running, t_1, 5)$

specifies that the velocity of "running" at time point t_1 is 5. The relation

$displacement(P_1, P_2, B, P_3)$

is true if and only if the displacement due to B over the interval (P_1, P_2) is P_3. For example, ran a kilometer over the interval (t_1, t_2) is written as:

$displacement(t_1, t_2, running, 1)$.

The relation $acceleration(B, P_1, P_2)$ is true if and only if the acceleration of B at time point P_1 is P_2. For example, the following

$acceleration(running, t_1, 2)$

specifies that the acceleration of "running" at time t_1 is 2.

4 GCH's syntax and semantics

4.1 Syntax

GCH's syntax is defined in terms of two first order logics called T and W. To represent time we use T, an ordinary first order logic suitable for describing simple arithmetic operations and equality over the reals. T's constants are

$\{t_1, t_2, \ldots\} \cup \mathbb{R}$

its variables are

$\{T_1, T_2, \ldots\}$

its functions are

$\{+, -, \times\}$

and its predicates are

$\{<, \leq, >, \geq, =\}$.

Terms and formulas are defined in the normal way. The following is an example of a formula:

$\forall T_1 \ . \quad 1 \times T_1 = T_1$.

W is an ordinary non-temporal first order logic suitable for describing a world at a single point in time. Therefore the relations in W are ones that can be true or false at a point in time. For example "red(block1)" is acceptable but not "run-a-mile(John)" because it cannot be true at a point. Note that the relations of W have no temporal arguments. For example, "red(2pm,block1)" is not acceptable. For the sake of exposition, assume that the variables of the language W are $\{W_1, W_2, \ldots\}$. For example,

$\forall W_1 \ . \quad red(W_1) \rightarrow above(W_1, block1)$.

W varies from domain to domain and therefore it is impossible to specify particular predicate and function symbols a priori.

GCH has two sorts of terms: *temporal* and *non-temporal*. The *temporal* terms consist of the terms from T. *Non-temporal* terms are defined inductively as follows

- The atoms of W are *non-temporal* terms.
- Each member of V = $\{V_1, V_2, \ldots\}$ is a *non-temporal* term.
- If β is a *non-temporal* term, then $\neg\beta$ is a *non-temporal* term.

The formulas of GCH are defined inductively as follows

- The atoms of T are formulas.
- If π is a *temporal* term and β is a *non-temporal* term then $true(\pi, \beta)$ is a formula.
- If π_1, π_2, π_3 are *temporal* terms and β is a *non-temporal* term then

  $$integral(\pi_1, \pi_2, \beta, \pi_3)$$

  is a formula.
- If β is a *non-temporal* term and π_1, π_2, π_3 are *temporal* terms, then

  $$velocity(\beta, \pi_1, \pi_2),$$

  $$displacement(\pi_1, \pi_2, \beta, \pi_3),$$

  and

  $$acceleration(\beta, \pi_1, \pi_2)$$

  are formulas of GCH.
- If ϕ_1, ϕ_2 are formulas of GCH, then $(\phi_1 \wedge \phi_2)$, $(\phi_1 \vee \phi_2)$, $(\phi_1 \rightarrow \phi_2)$, $(\neg\phi_1)$ are formulas of GCH.
- If ϕ is a formula of GCH, and z is a variable from T or a variable from W or an element of V, then $(\forall z \, . \, \phi)$, $(\exists z \, . \, \phi)$ are formulas of GCH.

4.2 Semantics

A formal semantics for GCH appears in [Trudel 1989; Trudel 1990]. They are not included here because of space limitations.

4.3 Axiomatization of the *true* and *integral* relations

The axioms for the *true* and *integral* relations are presented in figure 1.

The first axiom in figure 1 captures the property that the value of the integral at a point is zero.

The second axiom captures the following additive property of the Riemann integral:

$$\int_{T_1}^{T_2} f(x)dx \quad + \quad \int_{T_2}^{T_3} f(x)dx \quad = \quad \int_{T_1}^{T_3} f(x)dx.$$

The third axiom places bounds on the value of the integral. Note that the consequent also captures the restriction that $T_1 \leq T_2$.

The fourth axiom captures the property that a *non-temporal* term is true throughout an interval iff the value of its integral equals the length of the interval. Note that the left hand side of the fourth axiom makes no commitment at the endpoints. The truth value of $integral(T_1, T_2, V_1, T_2 - T_1)$ is independent of the truth values of $true(T_1, V_1)$ and $true(T_2, V_1)$. This is a useful property. For example, if we have:

$$integral(t_1, t_2, running, t_2 - t_1) \;\; \wedge$$

$$integral(t_2, t_3, \neg running, t_3 - t_2)$$

then the fourth axiom cannot be used to determine the truth value of

$$true(t_2, running).$$

The last four axioms in figure 1 deal with negation inside the *true* and *integral* relations. The fifth and sixth axioms capture the fact that double negation inside the *true* and *integral* relations cancels.

The last two axioms are for removing single negation from inside the *true* and *integral* relations. Note there is not an *integral* version of the seventh axiom, i.e.,

$$\forall T_1, T_2, T_3, V_1 \, . \, integral(T_1, T_2, \neg V_1, T_3) \;\; \leftrightarrow$$
$$\neg integral(T_1, T_2, V_1, T_3)$$

is not an axiom. A counterexample to the above is the fact that the following is consistent:

$$integral(0, 1, \neg running, 0.5) \;\; \wedge$$

$$integral(0, 1, running, 0.5).$$

Using the last four axioms, we can eliminate negation from inside the *true* and *integral* relations. For example, an equivalent form of

$$true(1, \neg p) \;\; \wedge \;\; true(1, \neg\neg q) \; \wedge$$

$$integral(1, 10, \neg a, 5) \; \wedge$$

$$integral(1, 10, \neg\neg b, 5)$$

is

$$\neg true(1, p) \;\; \wedge \;\; true(1, q) \; \wedge$$

$$integral(1, 10, a, 10 - 1 - 5) \; \wedge$$

$$integral(1, 10, b, 5).$$

5 Examples

In this section we present various examples using GCH.

5.1 Swimming

Anybody that has taken a swimming lesson knows that we must wait at least two hours after eating, before swimming:

$$\forall T_1 \; . \;\; true(T_1, swimming) \;\; \rightarrow$$
$$integral(T_1 - 2, T_1, \neg eating, 2).$$

The above says that if *swimming* is true at some point in time T_1 then $\neg$*eating* is true at each point in the two hour interval which ends at T_1.

5.2 Running

Assume the velocity of running is 2 miles per time unit:

$$\forall T_1 \; . \;\; velocity(running, T_1, 2)$$

Ran a mile over the interval (t_1, t_2) is written as:

$$displacement(t_1, t_2, running, 1).$$

Note that in this example, the truth value of *running* and the displacement due to *running* are related as follows:

1. $\forall T_1, V_1 \,.\, integral(T_1, T_1, V_1, 0)$.
2. $\forall T_1, T_2, T_3, T_4, T_5, V_1 \,.\, [integral(T_1, T_2, V_1, T_4) \wedge integral(T_2, T_3, V_1, T_5)] \rightarrow integral(T_1, T_3, V_1, T_4 + T_5)$.
3. $\forall T_1, T_2, T_3, V_1 \,.\, integral(T_1, T_2, V_1, T_3) \rightarrow 0 \le T_3 \le (T_2 - T_1)$.
4. $\forall T_1, T_2, V_1 \,.\, [\forall T_3 \,.\, T_1 < T_3 < T_2 \rightarrow true(T_3, V_1)] \leftrightarrow integral(T_1, T_2, V_1, T_2 - T_1)$.
5. $\forall T_1, T_2, T_3, V_1 \,.\, integral(T_1, T_2, \neg\neg V_1, T_3) \leftrightarrow integral(T_1, T_2, V_1, T_3)$.
6. $\forall T_1, V_1 \,.\, true(T_1, \neg\neg V_1) \leftrightarrow true(T_1, V_1)$.
7. $\forall T_1, V_1 \,.\, true(T_1, \neg V_1) \leftrightarrow \neg true(T_1, V_1)$.
8. $\forall T_1, T_2, T_3, V_1 \,.\, integral(T_1, T_2, \neg V_1, T_3) \leftrightarrow integral(T_1, T_2, V_1, T_2 - T_1 - T_3)$.

Figure 1: Axioms for the *true* and *integral* relations

$$\forall T_1, T_2, T_3 \,.\, displacement(T_1, T_2, running, 2 \times T_3) \leftrightarrow integral(T_1, T_2, running, T_3).$$

We can also represent ran a mile over some interval, say (5,40), if the velocity of *running* is unknown. In this case we simply write

$$displacement(5, 40, running, 1)$$

and do not specify the *velocity* relation.

5.3 Table lifting

There is a table resting on the floor with one agent at each end. An agent can lift his end of the table independently of the other. On the table is a full cup of coffee. If the difference in height between the left and right sides of the table is $> t_4$, then the coffee spills. The problem is to represent the following situation: the agents lift the table a height t_3 in the interval (t_1, t_2) without spilling any coffee. The solution is shown in figure 2. The first two formulas specify that the right and left ends of the table are lifted a height t_3 in the interval (t_1, t_2). The third formula specifies that the table must remain within t_4 units of level. This formula says that for each point T_5 in the interval (t_1, t_2), the difference in height at the point T_5 is $\le t_4$.

5.4 Ball in air

Assume we throw a ball in the air at time 0 and it doesn't hit the ground until time 6. The ball reaches the apex of its trajectory at time t_1 where $0 < t_1 < 6$. The vertical velocity of the ball is zero at time t_1 and non-zero everywhere else in the interval (0,6). The GCH representation of the example is shown in figure 3. Note that we represented the above example without any knowledge of the velocity function associated with the ball.

5.5 Road runner and coyote

In [1989], Sandewall presents an example of a ball rolling a short distance and then falling in a shaft. Here we simplify and modify the example slightly. In every "Road Runner" cartoon, the coyote ends up jumping off a cliff. Such a leap is shown in figure 4. At time 0, the healthy coyote is at location (0,100) and begins running towards the cliff at a constant velocity of 40 units per time unit. A universal property of cartoon characters is that they can run past the edge of a cliff quite some distance before starting their downward descent. The coyote makes it to the point (160,100) before falling. At the point (160,100), he begins falling with an acceleration of −9.8.

The coyote reaches the point (160, 100) at time 4, and splatters on the ground at time t_1 (at which time a cloud of dust rises). t_1 is specified by the following:

$$displacement(4, t_1, falling, 100).$$

The coyote example is represented as follows in GCH. Before time 4:

$$integral(0, 4, running, 4).$$
$$integral(0, 4, falling, 0).$$
$$\forall T_1 \,.\, 0 < T_1 < 4 \rightarrow velocity(running, T_1, 40).$$
$$\forall T_1 \,.\, 0 < T_1 < 4 \rightarrow velocity(falling, T_1, 0).$$

After time 4:

$$integral(4, t_1, running, 0).$$
$$integral(4, t_1, falling, t_1 - 4).$$
$$\forall T_1 \,.\, 4 < T_1 < t_1 \rightarrow velocity(running, T_1, 0).$$
$$\forall T_1 \,.\, 4 < T_1 < t_1 \rightarrow velocity(falling, T_1, -9.8 \times (T_1 - 4)).$$

6 Conclusions

In this paper we presented a first order logic for representing temporal domains called GCH. GCH uses a new approach for representing interval based information. Instead of directly associating this type of information with an interval, we use the Riemann integral. Capturing the Riemann integral in the logic gives us an intuitive, simple, and precise method of representing interval based information. Also, this approach easily deals with both qualitative and quantitative interval based information.

$displacement(t_1, t_2, lift\_right\_side, t_3).$

$displacement(t_1, t_2, lift\_left\_side, t_3).$

$$\forall T_5, T_6, T_7 \,.\;\; t_1 \leq T_5 \leq t_2 \;\wedge\; displacement(t_1, T_5, lift\_right\_side, T_6) \;\wedge\; displacement(t_1, T_5, lift\_left\_side, T_7) \;\rightarrow\; -t_4 \leq (T_6 - T_7) \leq t_4.$$

Figure 2: Table lifting solution

$integral(0, 6, ball\_in\_air, 6).$

$0 < t_1 < 6.$

$velocity(ball\_in\_air, t_1, 0).$

$$\forall T_1 \,.\; [0 < T_1 < 6 \;\wedge\; T_1 \neq t_1] \;\rightarrow\; [\exists T_2 \,.\; velocity(ball\_in\_air, T_1, T_2) \;\wedge\; T_2 \neq 0].$$

Figure 3: Ball in air solution

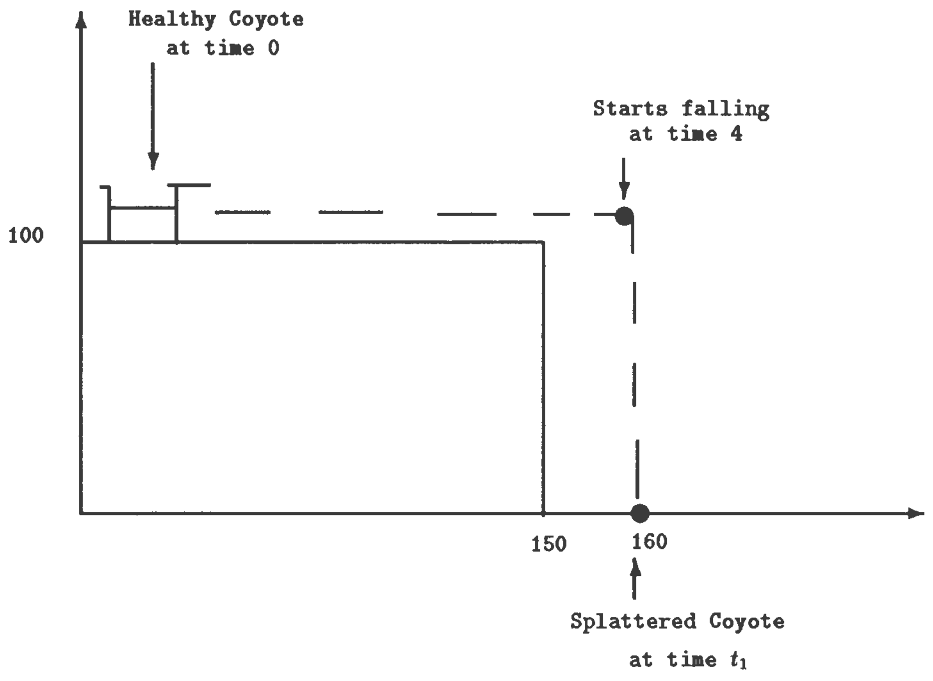


Figure 4: Coyote example

Acknowledgements

This work would not have been possible without the help of Fahiem Bacchus, Scott Goodwin, Eric Neufeld, David Poole, and the members of the Logic Programming and AI Group at the University of Waterloo. Thanks also to Sam and Max for valuable discussions on wffs.

References

[Allen, 1984] J.F. Allen, *Towards a General Theory of Action and Time.* Artificial Intelligence **23** (2), 123–154, 1984.

[McDermott, 1982] D. V. McDermott, *A temporal logic for reasoning about processes and plans.* Cognitive Science **6**, 101–155, 1982.

[Sandewall, 1989] E. Sandewall, *Combining logic and differential equations for describing real-world systems.* KR89, Toronto, Canada, 412–420, 1989.

[Shoham, 1988] Y. Shoham, *Reasoning about Change.* The MIT Press, Massachusetts, 1988.

[Trudel, 1989] A. Trudel, *The interval representation problem.* Submitted to the International Journal of Intelligent Systems, special volume on temporal issues of AI, 1989.

[Trudel, 1990] A. Trudel, *Representing and reasoning about a dynamic world.* Phd thesis, in preparation, 1990.

Explaining Decision-Theoretic Choices*

David A. Klein
IBM Thomas J. Watson Research Center
PO Box 704
Yorktown Heights, NY 10598
U.S.A.

Edward H. Shortliffe
Section on Medical Informatics
Stanford University School of Medicine
Stanford, CA 94305
U.S.A.

Abstract

We present strategies for explaining decision-theoretic choices, and we describe the role of these strategies in *Interpretive Value Analysis*, our broader framework for building expert systems that choose among alternatives. We demonstrate the explanation strategies with implemented examples in the domains of marketing, process control, and medicine. Although previous approaches to modeling choices in expert systems generally have sacrificed formal specification for transparent operation, our approach suggests that knowledge engineers can retain the benefits of a formal foundation without compromising intuitive explanation.

1 Introduction

Choices among competing alternatives arise in diverse expert-system domains, from medicine (e.g., choosing among alternative treatments) to process control (e.g., choosing among alternative control actions). A model for choosing among alternatives generally accepts as input a set of objective data and subjective judgments that characterizes a choice, and produces as output a recommended alternative. Such a model can be the sole reasoning machinery of an expert system that assists users with decisions, or can be integrated with other reasoning modules (e.g., conflict resolution in rule-based systems).

Previous approaches to modeling choices in expert systems often have sacrificed general, formal specification for transparent operation: Clancey [1984], in his description of MYCIN's therapy-planning model, for example, recounts that, "to formulate judgments that could be provided *by* physicians and would appear familiar *to* them, we decided not to use mathematical methods such as evaluation polynomials," but that, "relating decisions is difficult because they require some representation of what the heuristics mean." Similarly, Rennels *et al.* [1987] note that a formal decision-theoretic model "may obscure the salient features of the problem, trading off an ability to explain a choice in intuitive terms in favor of achieving a more powerful, generalized characterization of the problem."

In this paper, we present strategies for explaining decision-theoretic choices, and we describe the role of these strategies in *Interpretive Value Analysis (IVA)*, a general framework for modeling choices in expert systems that is at once formal *and* transparent. We demonstrate the explanation strategies with implemented examples in the domains of marketing, process control, and medicine. In contrast with previous approaches to modeling choices in expert systems, our methodology suggests that knowledge engineers can retain the benefits of a formal foundation without compromising intuitive explanation.

The paper is organized as follows. Section 2 provides an overview of *Multiattribute Value Theory*, the discipline of decision theory on which IVA is based. Section 3 outlines the structure of IVA to provide a context for describing the explanation strategies reported in Section 4. Section 5 provides examples of IVA-based explanation facilities that draw on the strategies of Section 4. We elucidate the pragmatic advantages of IVA's decision-theoretic model in Section 6, and we evaluate IVA's explanations in the context of previous approaches in AI. Section 7 provides a summary of the paper, and reviews our conclusions.

2 Background: Multiattribute Value Theory

Multiattribute Value Theory addresses the problem of modeling **multiattribute choices under certainty**, in which multiple, often *mutually competitive* objectives

* This work was supported in part by the IBM Corporation, Grant LM05208 from the National Library of Medicine, the IBM Graduate Fellowship Program, and the NASA Graduate Student Researcher's Program.

underlie a choice among alternatives, and the outcomes of alternatives are known with virtual certainty. Consider, for example, a computer-complex manager's choice among competing actions for purging a dataset that is awaiting service on a disabled printer, and is overloading the system queue. The manager can remove the dataset from the queue by executing any of several actions, such as deleting the dataset (DELETE), or transferring the dataset to the user's private disk space (DASD). The manager's choice among such actions involves tradeoffs among competing objectives: Executing DELETE, for example, clears the queue more quickly than many alternative actions, but causes the dataset's owner greater inconvenience than do these other actions. The outcomes of dataset-purging actions are known with virtual certainty: Executing DELETE almost always results in the instantaneous deletion of a dataset, for example. The models of Multiattribute Value Theory potentially are useful for reasoning about multiattribute choices under certainty in expert systems, because these models are supported by a formal theory and by a well-developed methodology.[1]

The problem of modeling multiattribute choices under certainty can be stated more formally as follows.[2] Let a designate a feasible **alternative,** such as an action for purging a dataset, and let A denote the set of all such alternatives. With each act $a \in A$, we associate n indices of value, $X_1(a), \dots, X_n(a)$, that reflect our objectives; an **objective** is a reason to care about the selection of one alternative rather than another, such as the desire to minimize the time it takes to remove a dataset from the queue. We describe the degree to which our objectives are satisfied in the context of **attributes** of alternatives, such as minutes elapsed in purging a dataset. Roughly, the decision maker's problem is to choose $a \in A$ such that he will be happiest with the payoff $X_1(a), \dots, X_n(a)$. Thus, we need a mechanism v that combines $X_1(a), \dots, X_n(a)$ into a scalar index of preferability or value. We refer to the function v as the **value function.** Given v, we choose $a \in A$ such that v is maximized.

The appropriate form for the value function depends on the relationship among the decision maker's objectives; the form employed in most practical applications is the **Additive Multiattribute Value Function (AMVF):**[3]

$$v(a) = v(x_1, \dots, x_n) = \sum_{i=1}^{n} w_i v_i(x_i)$$

1. Each $a \in A$ is represented by a vector of **attribute values** $(x_1, \dots, x_n)$. The vector representing DELETE, for example, includes the value \$0.00 for attribute *additional cost,* because deleting a dataset creates no additional material cost beyond the user's original print request.
2. v_i is the **component value function** for attribute i, with $v_i(\text{worst } x_i) = 0$, $v_i(\text{best } x_i) = 1$, and $0 \le v_i(x_i) \le 1$ for all x_i. The component value functions express the relative desirability of various levels of their respective attributes; for example, the component value function for attribute *additional cost* assigns 0 to the action(s) of greatest additional cost and assigns 1 to the action(s) of least additional cost.
3. w_i is the **weight** for attribute i, $0 < w_i < 1$ and $\sum w_i = 1$. The weights indicate the relative importance of each attribute as it changes from its best to its worst value: A model of the preferences of a relatively cost-conscious manager, for example, would include a relatively high weight for attribute *additional cost.*

The construction of a value function can be facilitated by the employment of a **value tree,** which represents explicitly the decomposition of the user's overall objective (the root of the tree) into a hierarchically structured set of more detailed objectives. Figure 1 depicts a simple value tree for managing queue space.

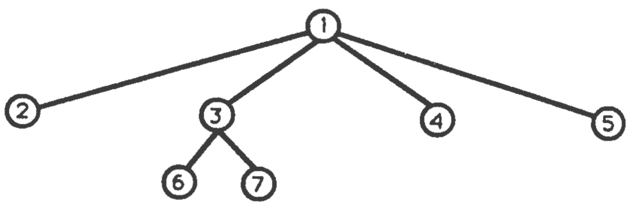


Figure 1: A simple value tree for managing queue space. The satisfaction of a nonleaf objective, such as *maximize user satisfaction* (node 3), depends on the satisfaction of its children. Leaf objectives, such as *minimize additional turnaround time* (6), are associated with attributes in the value function; the satisfaction of a leaf objective is represented by its associated component value function, as we described. Local weights on sibling objectives sum to 1 in every subtree. The weight for an attribute can be calculated by taking the product of the local weight of the associated leaf objective and the weights of its ancestors: The weight for *minimize additional turnaround time* (6), for example, is given by $w_6 w_3$.

1 We shall elaborate these advantages in Section 6. Langlotz [1989] describes related advantages in the context of modeling *single*-attribute choices under *un*certainty in expert systems.

2 Portions of this description are adapted from [Keeney and Raiffa, 1976].

3 Loosely speaking, the use of this form requires that the decision maker be able to express his preferences in each attribute independently of other attributes. Authors such as Keeney and Raiffa [1976] provide detailed specifications for constructing decision-theoretic models that reflect a variety of assumptions, and for verifying the satisfaction of such assumptions in practice.

The root of the tree (node 1) represents the overall objective to *maximize queue-space-management effectiveness*. This objective is an abstract representation of its children -- *minimize additional operator time* (2), *maximize user satisfaction* (3), *minimize additional cost* (4), and *minimize problem-resolution time* (5). *Maximize user satisfaction* (3) is, in turn, an abstraction of objectives *minimize additional turnaround time* (6) and *maximize form similarity* (7).

3 Interpretive Value Analysis

Interpretive Value Analysis (IVA) [Klein, 1989] is a framework for explaining and refining the AMVF. The design of IVA reflects empirical observations; interviews with both decision analysts and nonanalysts suggest an **interpretation** for the AMVF that provides a formal vocabulary of approximately 100 terms, or **interpretation concepts**, for talking about choices. The expression $w_i(v_i(a1_i) - v_i(a2_i))$, for example, can be interpreted intuitively as a measure of the strength or COMPELLINGNESS[4] of a reason or objective i for choosing alternative $a1 = (a1_1, ..., a1_n)$ rather than $a2 = (a2_1, ..., a2_n)$, as in the statement, "Additional cost is a compelling reason to prefer DELETE to EXPENSIVE.PRINTING." We show in [Klein, 1989] that $w_i(v_i(a1_i) - v_i(a2_i))$ is a well-formed expression with respect to the particular value-measurement scale on which IVA is based.[5] We provide analogous formal links between this version of value theory and interpretation concepts such as NOTABLY-IMPORTANT? ("Is additional operator time notably important in managing queue space?"). Such formal descriptions guarantee the consistency of explanations, because we restrict the content of explanations to interpretation concepts, all of which are defined with respect to the same value-theoretic model.

The interpretation concepts are the primitives of IVA's **explanation strategies**, which are designed to provide the user with sufficient insight into an AMVF's operation either (1) to become convinced that the chosen alternative is indeed preferred, or (2) to identify for correction a model parameter that deviates from his true preferences; IVA's **refinement strategies** [Klein and Shortliffe, 1990] facilitate such corrections, building on the explanation strategies.

Klein [1989] provides a detailed exposition of interpretation, explanation, and refinement under IVA. This paper focuses on IVA's explanation strategies.

4 Explanation Strategies

We provide in IVA explanation strategies that allow users to pose detailed queries about choices, to generate summary comparisons of choices, to observe the computation of choices in a step-by-step fashion, and to generate abstract descriptions of how choices are computed. We demonstrate selected strategies in the following sections, employing examples generated by **VIRTUS**,[6] an IVA-based shell that has been tested in the domains of marketing, process control, and medicine.

4.1 Generation of Responses to Detailed Queries

IVA's interpretation concepts provide a space of queries that can be made available to the user directly: VIRTUS, for example, includes an **interpretation-concept invocation** facility that permits users to view interpretation-concept results by specifying interpretation concepts and their operands (alternatives and objectives). In the following example, generated by a VIRTUS application for evaluating expert-system shells, VIRTUS paraphrases the user's mouse-and-menu query (**bold-face type**) and displays a textual response (*italics*):[7]

What is the relative quality of SHELL.B and SHELL.C with regard to development environment support?

SHELL.B provides somewhat better development environment support than SHELL.C.

In this simple example, the user requests the result returned by ABSTRACT-RELATIVE-QUALITY, an interpretation concept that maps the quantitative result [v(SHELL.B) − v(SHELL.C)] (with respect to objective *maximize development environment support*) to the qualitative value *somewhat better*. The user can invoke such interpretation concepts arbitrarily to conduct an intuitive interactive dialog with VIRTUS:

Which alternative is best with regard to overall value?

SHELL.B is the best with regard to overall value.

What is the relative quality of SHELL.B and SHELL.C with regard to overall value?

SHELL.B provides substantially better overall value than SHELL.C.

What are notably compelling reasons in the choice between SHELL.B and SHELL.C with regard to overall value?

Price provides the most compelling reason for the choice.

4 Throughout the paper, we display interpretation concepts in HYPHENATED-SMALL-CAPITAL-TYPEFACE.

5 IVA is based on the theory of *Measurable Multiattribute Value Functions* [Dyer and Sarin, 1979], which employs an interval scale of measurement for value. This scale lends meaning to statements such as "$a1$ is ***much better*** than $a2$," whereas classical utility theory [von Neumann and Morgenstern, 1947], for example, supports only ordinal statements, such as "$a1$ is ***better*** than $a2$."

6 VIRTUS is Latin for *value*, as in the value of an alternative in a choice among alternatives.

7 In formatting selected VIRTUS examples, we have made cosmetic corrections, such as capitalization and pluralization.

What is the relative quality of SHELL.B and SHELL.C with regard to development environment support?

SHELL.B provides somewhat better development environment support than SHELL.C.

What are the reasons for and against SHELL.B vs. SHELL.C with regard to development environment support?

Quality of documentation and quality of front end are reasons to prefer SHELL.B over SHELL.C with regard to development environment support. Reliability, interactive development facilities, and syntactic familiarity to data processing programmers are reasons to prefer SHELL.C over SHELL.B.

What factors are notably important in the determination of development environment support?

Quality of documentation is notably important in the determination of development environment support.

{And so on}

Although useful for exploring the details of a choice in an unrestricted fashion, interpretation-concept invocation burdens the user with coordinating VIRTUS's responses to achieve an understanding of a choice. The following explanation strategy imposes organization on such responses.

4.2 Generation of Summary Comparisons

Value-tree pruning and presentation strategies generate summary-level comparisons of particular pairs of alternatives. Generating a summary comparison involves *pruning* objectives from an explanation by (1) limiting the depth and breadth of the value tree in accordance with the alternatives of interest, and (2) generating an explanation that reflects the remains of the tree.

Vertical-pruning strategies determine a desirable level of abstraction for talking about objectives (in the context of comparing a particular pair of alternatives) by pruning detailed objectives that can be summarized by a higher-level objective; these strategies limit the depth of the value tree by eliminating uninteresting subtrees. The following algorithm accepts as input a value tree of arbitrary depth with root *root*, and two alternatives *a1* and *a2*, and produces as output a population of interesting objectives, if any exist, and *root* otherwise.

```
1. Let o = PARENT(deepest leaf)
2. If CARDINAL-TRADEOFFS?(a1 a2 CHILDREN(o)),
   then mark o as deleted
   otherwise, mark CHILDREN(o) as deleted
3. If o = root, then return the remains of the tree;
   otherwise, go to 1.
```

The interpretation concept CARDINAL-TRADEOFFS? is employed in the algorithm to identify interesting subtrees, returning T only when *a1* is strictly better than *a2* with respect to at least one objective, and is strictly worse with respect to at least one objective. On each iteration, the algorithm prunes either the current node in the tree (if the node's children are interesting), or the current node's children (if the children are uninteresting). For example, whenever *a1* and *a2* are EQUIVALENT? with respect to all the objectives in a subtree (one instance of CARDINAL-TRADEOFFS? = NIL), the algorithm prunes these objectives from the explanation and retains their parent, because the effects of these objectives can be summarized intuitively by their parent. Figure 2 shows a vertically-pruned value tree from a VIRTUS application that evaluates *randomized clinical trials (RCTs)* -- studies that compare the relative effectiveness of treatments by randomly assigning alternate therapies to subjects and observing the effects.

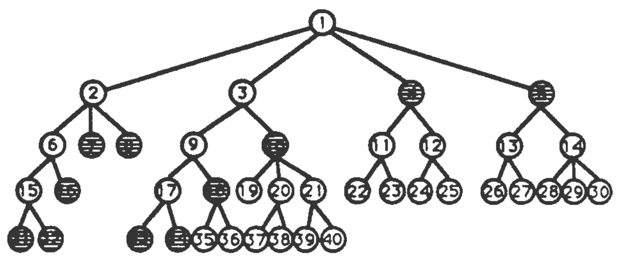


Figure 2: Vertical pruning. Comparing two particular alternatives RCT.1 and RCT.2, vertical pruning retains the following objectives (shaded in the figure), which vary in abstraction: *adjustments due to subgroups* (31), *subgroup analysis plan* (32), *stopping appropriateness* (16), *P-value* (7), *statistical techniques* (8), *distance of assigner from patient* (33), *quality of actual treatment assignment* (34), *blinded assignment design* (18), *equivalence at study outset* (10), *equivalence of care* (4), and *endpoint assessment* (5). Other objectives are pruned from the associated explanation.

This vertically pruned tree provides the basis for the following explanation:

Equivalence of care, equivalence at study outset, endpoint assessment, distance of assigner from patient, and adjustments due to subgroups are reasons to prefer RCT.1 over RCT.2 with regard to credibility. Quality of actual treatment assignment, blinded assignment design, stopping appropriateness, and subgroup analysis plan are reasons to prefer RCT.2 over RCT.1. P-value and statistical techniques do not at all impact the choice between RCT.1 and RCT.2.

Although more terse than an exhaustive display of value-tree leaves, this explanation is somewhat verbose, because limiting only the depth of the value tree still allows for arbitrary breadth.

The breadth of the value tree can be reduced by **horizontal-pruning strategies,** which eliminate objectives at a particular level. IVA includes several horizontal-pruning strategies, which vary in computational complexity and effectiveness. One inexpensive strategy, for example, simply selects the *n* most compelling objectives arguing for and against the preference of one alternative over another; this strategy is somewhat arbitrary, however, because we lack intuitive justification for omitting objective $n+1$ in the COMPELLINGNESS ranking whenever that objective differs only infinitesimally in COMPELLINGNESS from the nth. A more effective (but expensive) IVA strategy selects *clusters* of objectives for inclusion in explanations, based on a statistical interpretation concept that identifies outlying COMPELLINGNESS values in a population.

Combining vertical pruning with horizontal pruning yields a powerful capability for generating summary comparisons: Vertical pruning produces an appropriately abstract population of objectives, which is reduced further by horizontal pruning, as shown in Figure 3.

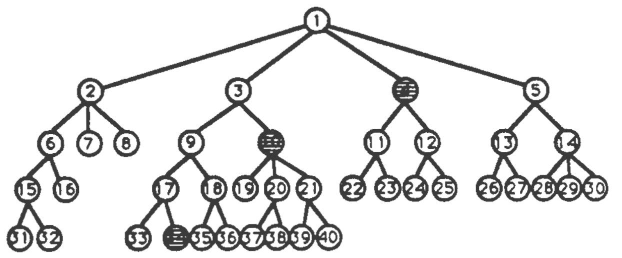


Figure 3: Combined vertical and horizontal pruning. A horizontal pruning strategy selects the notably compelling objectives (shaded) from the vertically pruned tree of Figure 2, retaining only *quality of actual treatment assignment* (34), *equivalence at study outset* (10), and *equivalence of care* (4).

The population of objectives retained in Figure 3, embellished with ABSTRACT-RELATIVE-QUALITY (Section 4.1), provides the basis for generating the following summary:

> *RCT.1 is somewhat better than RCT.2. Compelling reasons to prefer RCT.2, such as quality of actual treatment assignment, are outweighed by considerations of equivalence of care and equivalence at study outset, along with less compelling reasons that recommend RCT.1.*

Such explanations capture the most compelling reasons for preferring a particular alternative to another, at an appropriate level of abstraction. Whenever a user finds a summary-level comparison of two alternatives unconvincing, however, he may require a more detailed comparison: In the following section, we present strategies that demonstrate intuitively the detailed relationships between the user's primitive inputs (parameter values) and the model's results.

4.3 Generation of Detailed Comparisons

IVA's **difference-function-traversal strategies** generate intuitive detailed comparisons of alternatives. These comparisons demonstrate the relationships between a model's results and parameter values by explaining the step-by-step computation of $[v(a1) - v(a2)]$, the overall RELATIVE-QUALITY of two particular alternatives $a1$ and a2. Difference-function-traversal strategies reflect a decompositional approach to explanation that involves (1) representing explicitly the precedence of operations that combine parameters, (2) providing an intuitive explanation for each such operation, and (3) designing control procedures that concatenate these explanations according to the operation-precedence representation. An analogous approach has been employed to support explanation in systems that model the behavior of physical devices [De Kleer and Brown, 1984].

Under IVA, generating a step-by-step comparison of a specified pair of alternatives involves traversing a **difference function** h, which we developed to reflect elements of our subjects' explanations (Section 3). More specifically, a difference function captures interactions among AMVF parameters (Section 2) for a pair of alternatives ($a1$,$a2$), and reflects the hierarchical structure of the value tree: The difference function corresponding to the tree of Figure 1, for example, is

$$\begin{aligned} h(a1,a2) = {} & [w_2(v_2(a1_2) - v_2(a2_2))] \\ & + w_3[w_6(v_6(a1_6) - v_6(a2_6)) + w_7(v_7(a1_7) - v_7(a2_7))] \\ & + [w_4(v_4(a1_4) - v_4(a2_4))] \\ & + [w_5(v_5(a1_5) - v_5(a2_5))] \end{aligned}$$

The step-by-step execution of this function is represented explicitly in the **topology** shown in Figure 4.

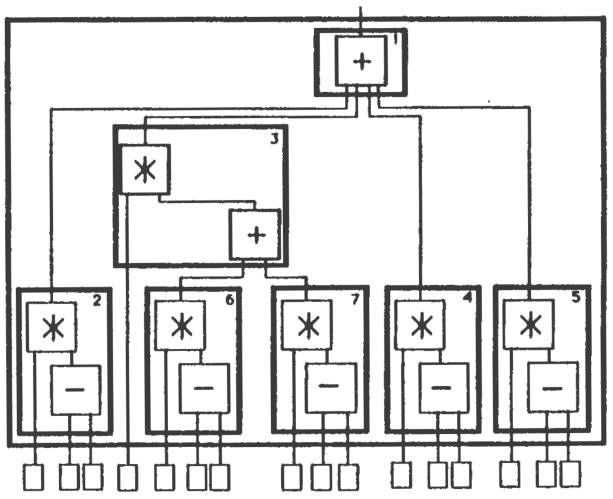


Figure 4: The topology for the value tree of Figure 1. The subtopologies framed by boxes correspond to objectives in the value tree, as labeled. For uniformity, parameters are represented as operations that return (user-specified) constants.

We associate each arithmetic operation in the topology with an **operation explainer** that generates an intuitive explanation of the relationship between its operands and its result. Addition in the difference function, for example, accepts as input a set of COMPELLINGNESS values for objectives that argue for and against a choice between two particular alternatives, and produces as output the RELATIVE-QUALITY of these alternatives; VIRTUS's addition explainer provides an intuitive summary of this operation, such as the following explanation (from our VIRTUS application for managing queue space):

DASD provides infinitesimally better overall queue space management effectiveness than EXPENSIVE.PRINTING. While overall user satisfaction provides a compelling reason to prefer EXPENSIVE.PRINTING, this is outweighed by considerations of additional cost, along with other less compelling reasons, that provide motivation for preferring DASD.

Control strategies generate explanations by traversing a topology and concatenating the explanations generated by operation explainers. **Backward chaining** over a topology provides an account of how the final result is derived from intermediate results, and of how intermediate results are, in turn, derived from primitive inputs. The following explanation fragment, for example, is produced by backward chaining over a multiplication explainer and its connected subtraction explainer for leaf objective *minimize additional cost* (box (4) in Figure 4) in a comparison of alternatives DASD and EXPENSIVE.PRINTING:

$\boxed{*}$ *Additional cost is a compelling factor favoring DASD over EXPENSIVE.PRINTING. While additional cost is not notably important in determining overall queue space management effectiveness, DASD provides sufficiently different additional cost from EXPENSIVE.PRINTING, relative to other factors, to make additional cost a notably compelling factor in this particular decision.*

$\boxed{-}$ *DASD provides substantially better additional cost than EXPENSIVE.PRINTING. DASD provides notably good additional cost in the context of all available alternatives. EXPENSIVE.PRINTING provides notably poor additional cost in the context of all available alternatives.*

In this example, the subtraction explainer clarifies the explanation produced by the multiplication explainer, providing additional detail regarding the meaning of "sufficiently different."

In a similar fashion, **forward chaining** provides an account of how a particular parameter participates in the determination of an intermediate result, which in turn ultimately affects the final result. **Interactive chaining** permits the user to forward and backward chain over a topology in an incremental, interactive fashion. Klein [1989] provides a more detailed exposition of strategies for difference-function traversal.

5 Construction of Explanation Facilities

The modularity of IVA's explanation strategies permits flexibility in designing explanation facilities. The implementation of such facilities might vary, for example, with the designs of operation explainers, with the particular horizontal-pruning strategies that are employed, and more generally, with the classes of strategies that are included.

In addition, knowledge engineers can integrate IVA's explanation strategies to implement common explanation commands, such as WHY. For example, an engineer can provide an explanation module that generates a particular response to WHY (e.g., the summary comparison of Section 4.2); alternatively, the engineer might implement a menu-driven facility that permits the user to select among multiple interpretations of WHY, as in the following dialog:

DASD is the best with regard to overall queue space management effectiveness.

> **WHY**

By WHY, which of the following interpretations do you mean?

1. *What is notably good about DASD?*
2. *How does DASD compare with the next best alternative, EXPENSIVE.PRINTING?*
3. *What are DASD's close contenders?*
4. *How does DASD compare with its close contenders?*
5. *List the alternatives that are not as good as DASD.*
6. *...*

> **1**

DASD provides notably good additional cost and problem resolution time.

Questions such as (1), (3), and (5) can be answered directly by interpretation concepts, whereas responses to questions such as (2) and (4) might employ value-tree pruning and presentation or difference-function traversal.

IVA's explanation strategies also can be integrated to generate intuitive reports about decisions. VIRTUS, for example, generates reports that are organized as follows:

1. Problem Statement
2. Solution Summary
3. The Decision-Making Process
4. Brief Analysis of the Decision
5. Brief Analysis of Close Contenders
6. Detailed Analysis of Close Contenders

These reports draw on a variety of IVA's interpretation concepts and explanation strategies: The "Solution Summary," for example, includes a reference to the concept SIMILAR-ALTERNATIVES to identify the best alternative's close contenders; these contenders are compared with the best alternative by value-tree pruning and presentation (in "Brief Analysis of the Decision"). The alternatives are compared again by a backward-chaining, difference-function-traversal strategy that prunes uninteresting operation-explainer outputs (in "Brief Analysis of Close Contenders"), and once again by exhaustive, backward-chaining difference-function traversal (in "Detailed Analysis of Close Contenders"). [Klein, 1989] provides a complete example of such a report.[8]

6 Relationship to Previous Approaches in AI

IVA provides potential advantages over previous AI systems that model multiattribute choices under certainty, with respect to formal specification and to transparent operation.[9] The following analytical observations derive additional support from informal evaluations of VIRTUS by experts in our three domains, although these evaluations naturally lack the persuasiveness of more formal, statistically significant studies.

First, IVA's value-theoretic foundation permits the knowledge engineer to avoid some of the reasoning-related pitfalls of previous AI systems, such as reliance on case-specific machinery. QBKG [Berliner and Ackley, 1982], for example, a backgammon-playing system, employs a value-tree-like structure that allows multiple parents, an AMVF-like sum for computing some evaluations, and special expressions for computing other evaluations. QBKG's designers may have included these special expressions to offset the effects of objectives with multiple parents, which violate the independence constraints of the AMVF; the same strategy was implemented using special rules by the architects of REFEREE [Haggerty, 1984], the original EMYCIN implementation of our medical VIRTUS application. Reliance on such special expressions or rules to compensate for preferential dependence can result in a system that produces inconsistent choices. Because IVA is based on value theory, which specifies explicitly the required relationships among objectives, knowledge engineers are provided with guidelines for avoiding the inclusion of such case-specific machinery.[10] More generally, IVA's value-theoretic foundation provides confidence in the results of applications whenever the knowledge engineer agrees with the axioms of value theory and observes that these axioms are not violated systematically during problem structuring.

Second, IVA provides potential transparency-related advantages over previous AI systems for modeling multiattribute choices under certainty. Previous explanation systems generally have reflected domain-specific assumptions about the volume of objectives that underlie choices: BLAH [Weiner, 1980], for example, displays exhaustively the objectives that a user specifies, whereas IVA's combined vertical- and horizontal-pruning algorithms yield succinct summaries for value trees of arbitrary size. IVA also explains richer value-based relationships than do some previous systems; MYCIN's therapy-selection algorithm [Clancey, 1984], for instance, does not represent (or explain) tradeoffs among objectives explicitly, whereas this information is available for explanation under IVA.

These advantages are tempered, of course, by the significant cost of constructing a value function.

7 Summary and Conclusions

We presented strategies for explaining decision-theoretic choices, and we described the role of these strategies in IVA, a formal and transparent framework for modeling choices in expert systems. IVA's interpretation concepts provide an internally consistent vocabulary, based on value theory, for talking about choices. These interpretation concepts play the role of primitives in IVA's explanation strategies, which support detailed queries about choices, summary comparisons of alternatives, detailed comparisons of alternatives, and abstract descriptions of the computation of choices. Although previous approaches to modeling choices in expert systems generally have sacrificed formal specification for transparent operation, our approach suggests that knowledge engineers can retain the benefits of a formal foundation without compromising intuitive explanation.

Acknowledgments

Leslie Perreault and Rich Pelavin provided helpful comments on earlier drafts of this paper. Curt Langlotz, Martin Weber, and Tim Finin provided many useful discussions.

References

[Berliner and Ackley, 1982] Berliner, H., Ackley, D., The QBKG system: Generating explanations from a

[8] Langlotz [1989] describes the generation of analogous reports in the context of explaining choices based on decision trees.

[9] This analysis needs to be interpreted in the appropriate context: The authors of previous systems generally were *not* seeking a formal and transparent foundation for modeling choices, as we are in this work.

[10] Keeney [1981], for example, provides methods for restructuring sets of objectives to eliminate dependencies that disallow the employment of the AMVF.

non-discrete knowledge representation. *Proceedings of AAAI*, 1982.

[Clancey, 1984] Clancey, W., Details of the revised therapy algorithm. In Buchanan, B., Shortliffe, E., eds., *Rule-based Expert Systems*, Addison-Wesley, 1984.

[De Kleer and Brown, 1984] De Kleer, J., Brown, J., A qualitative physics based on confluences. *Artificial Intelligence 24*, 1984.

[Dyer and Sarin, 1979] Dyer, J., Sarin, R., Measurable multiattribute value functions. *Operations Research 27*, 1979.

[Haggerty, 1984] Haggerty, J., REFEREE and RULECRITIC: Two prototypes for assessing the quality of a medical paper. MS thesis, Report HPP-84-49, Knowledge Systems Laboratory, Stanford University, 1984.

[Keeney and Raiffa, 1976] Keeney, R., Raiffa, H., *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, Wiley, 1976.

[Keeney, 1981] Keeney, R., Analysis of preference dependencies among objectives. *Operations Research 29*, 1981.

[Klein, 1989] Klein, D., Interpretive value analysis. PhD thesis; available as RC #15278, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1989.

[Klein and Shortliffe, 1990] Klein, D., Shortliffe, E., Interactive diagnosis and repair of decision-theoretic models. Working Paper KSL-90-19, Section on Medical Informatics, Knowledge Systems Laboratory, Stanford University, 1990.

[Langlotz, 1989] Langlotz, C., A decision-theoretic approach to heuristic planning. PhD thesis, Section on Medical Informatics, Stanford University, 1989.

[Rennels *et al.*, 1987] Rennels, G., Shortliffe, E., Miller, P., Choice and explanation in medical management: A multiattribute model of artificial intelligence approaches. *Medical Decision Making 1*, 1987.

[von Neumann and Morgenstern, 1947]
von Neumann, J., Morgenstern, O., *Theory of Games and Economic Behavior*, Princeton University Press, 1947.

[Weiner, 1980] Weiner, J., BLAH: A system which explains its reasoning. *Artificial Intelligence 15*, 1980.

A Semantics for a Class of Inheritance Networks

James P. Delgrande
School of Computing Science,
Simon Fraser University,
Burnaby, B.C.,
Canada V5A 1S6

Abstract

A difficulty with virtually all general theories of default reasoning is that they are undecidable or, in the propositional case, intractable. Research in tractable defeasible reasoning has centred around inheritance reasoning; however the majority of these approaches lack any firm semantic justification. This paper attempts to bridge this gap by presenting a formally justified, tractable, approach to inheritance reasoning. This is accomplished by basing inheritance reasoning on an extant system of default reasoning, that of default reasoning in the logic N [Del88]. A set of syntactically-restricted strict and defeasible conditional assertions, expressed in N, is translated into a mathematical structure, called the canonical structure. This structure provably represents the set of all models of these sentences in the logic N. It is shown that inheritance reasoning may be defined and efficiently carried out in this structure. Moreover the inheritance inferencing is *correct*, in that it is sound with respect to default reasoning in [Del88]. We obtain then a tractable inheritance reasoner, with semantic justification furnished by default reasoning in N. In the terminology of [THT87] we effectively obtain a credulous, downward inheritance system, wherein defeasible and strict links may be mixed arbitrarily; hence this work provides a semantics for such systems. Lastly we argue that the work presented here points out a fundamental limitation with the inheritance diagrams used to motivate and justify prior systems of defeasible reasoning.

1 Introduction

This paper continues work in a specific approach to default reasoning by presenting a formally justified, tractable, system of inheritance reasoning. In [Del87] an extension to classical first-order logic is developed for reasoning about default statements such as "ravens are black" and "birds fly". This work is extended in [Del88] so that one can make inferences of default properties by means of meta-theoretic considerations. This allows us to conclude for example that if Opus is an albino raven then by default Opus is not black. The logic then provides a formal system for reasoning about statements of defaults, while the extension to this work provides a means of concluding default properties of individuals. A difficulty with the approach, as with many similar approaches, is that it is not obvious how it may be reasonably implemented.

In this paper I consider how the approach may be extended to deal with a restricted form of defeasible reasoning, that of inheritance reasoning. We begin with a set of strict and defeasible conditional assertions; these assertions are translated into a particular mathematical structure which, in a precise and intuitive sense, represents the set of all models of these sentences in the logic of [Del87]. It is shown that inheritance reasoning may be efficiently carried out in this structure. Since the defeasible inferencing is *correct*, in that it is shown to be sound with respect to the approach of [Del88], the approach provides a principled approach to such reasoning. In the terminology of [THT87] we obtain a credulous, downward inheritance system, wherein defeasible and strict links (negated or not) may be mixed arbitrarily; hence this work also provides a semantics for such systems. The key point then is that the semantic theory is used to sanction a particular approach to inheritance of properties. Alternatively, the approach may be viewed as formalising one of the set of "intuitions" referred to in [THT87].

In the next section the underlying formal system is described and related work is discussed. The section 3 informally describes the approach for inheritance reasoning while section 4 provides the formal details. Section 5 discusses some examples and points out a limitation with diagrams for inheritance networks. The last section gives a brief conclusion.

2 Background and Related Work

In [Del87] a *conditional logic*, *N*, for representing and reasoning about default statements is presented. This logic consists of first-order logic augmented with a *variable conditional* operator, $\Rightarrow$, for representing default statements. The statement $A \Rightarrow B$ is read as "if A then normally B". Thus "birds fly" is represented as

$\forall x(Bird(x) \Rightarrow Fly(x))$. This approach is intended to capture the notion of defeasibility found in naive scientific theories. That is, in the same way that "CO_2 freezes at $-40°$" states that a quantity of CO_2 *would* freeze at $-40°$ if it were pure, if the measuring equipment didn't affect the substance, etc., so too would a bird fly if we allowed for exceptional circumstances such as having a broken wing, being a penguin, etc.

Truth in the resulting logic is based on a possible worlds semantics: the accessibility relation between worlds is defined so that from a particular world w one "sees" a sequence of successively "less exceptional" sets of worlds. Two worlds in the same set are mutually accessible and, for two worlds appearing in different sets, one is accessible from the other but not vice versa. $A \Rightarrow B$ is true just when B is true in the least exceptional worlds in which A is true. Thus for example, "birds normally fly" is true if, in the least exceptional worlds in which there are birds, birds fly. Thus, intuitively, we factor out exceptional circumstances such as being a penguin, having a broken wing, being tied down, etc., and then say that birds fly if they fly in such "unexceptional" circumstances. In [Del87] a proof theory and semantic theory is provided and soundness and completeness results are obtained.

However, the conditional operator in this logic cannot support modus ponens. This is because if birds normally fly and Opus is a bird, then it may not be the case that Opus flies: perhaps Opus is a penguin. On the other hand, if we know only that Opus is a bird, then it seems reasonable to draw the default conclusion that Opus flies. This problem is addressed in [Del88], where two formulations of default inferencing are developed. Both formulations rely on meta-theoretic considerations to sanction default inferences. These approaches are proven to be equivalent with respect to their respective set of default inferences; hence here I will refer to the second approach only.

This second approach is based on the observation that default reasoning generally requires some sort of (explicit or implicit) simplicity assumption, stating that unless exceptional conditions are known to hold, they are presumed to not hold. This provides us with a justified means for changing the conditional operator $\Rightarrow$ in some of the defaults to the material conditional $\supset$. Thus for example, consider where we know only that R is (contingently) true in the world being modeled, and that the default $R \Rightarrow Bl$ is true. If the world being modelled is as simple (or unexceptional) as consistently possible then this world is among the least exceptional worlds in which R is true; from the semantic theory $R \supset Bl$ is true at such worlds also, and so we can conclude Bl by default.

There may be more than one way that one can justifiably make such transformations. Any single such way yields a *maximal contingent extension*; a property follows by default in this approach if it is true in all such extensions. With respect to the complexity of this procedure, the propositional case is exponential and the first-order case is undecidable. Thus while the system arguably provides a principled characterisation of defeasible reasoning, it appears to be not amenable to direct implementation on a computer. This problem is addressed here for the case of inheritance reasoning: that is, where we have a set of defaults, necessary implications, and negations of defaults, where the antecedent and consequent of the conditionals are restricted to be conjunctions of (possibly-negated) primitive propositions. In [Del90] the problem of arbitrary conditionals is addressed.

There has of course been extensive work in AI concerning default and defeasible inference. We can divide this work into two broad areas. First there are the approaches where a formal system is developed to characterise some particular approach to default reasoning. This work includes [Rei80; MD80; Moo83; Del88]. A general difficulty with these approaches is that while they may adequately characterise a particular phenomenon, they are not obviously realisable in a computer program, or not obviously efficiently realisable.

More recently there has been a good deal of interest in inheritance networks of defeasible properties, arguably beginning with [ER83] and including [HTT87; THT87; HT88; Hau88; Gef88; Bou89]. A difficulty with many of these approaches is that they lack any semantic basis, and so appeal simply to intuitions for their justification.[1] Geffner [Gef88] provides a probabilistic account of inheritance, and so appeals to intuitions differing from those here. Boutilier [Bou89] presents an approach superficially similar to that presented here; indeed the underlying formal system he employs is, with respect to inheritance, essentially that used here. However strict links are not employed, nor are negated defaults. More importantly, defeasibility rests on the extra-theoretic notion of *preferred models* [Sho88]. In contrast, we incorporate the semantic theory of the underlying logic directly in order to justify defeasible inference.

3 The Approach: Intuitions

From the semantic theory of N we have that the accessibility relation between possible worlds yields a sequence of successively "less exceptional" sets of possible worlds. Any two worlds in one of these sets are mutually accessible, and so these sets represent equivalence classes with respect to accessibility. The conditional $A \Rightarrow B$ is true if in the least set of worlds where there is a world in which A is true, B is true at these worlds also. I will use the notation s_A to denote the simplest set of worlds in which A is true at some world in this set.[2] This means that $A \Rightarrow B$ is true if $A \supset B$ is true at every world in s_A. We know then that in the simplest world(s) in which A is true, B must also be true; however B could also be true at yet simpler worlds. I will write this last result as $s_B \leq s_A$.

Consider what this means for default reasoning. I will use the example "ravens are normally black" and "albino ravens are normally not black"; these statements are represented propositionally, as $R \Rightarrow Bl$ and $R \wedge Al \Rightarrow \neg Bl$.

[1]This point is particularly well developed in [THT87].

[2]This of course assumes that there is a world where A is true. I will be making such an existence assumption throughout this paper with any mention of entities such as s_A.

We know also from classical logic that $R \wedge Al \supset R$. Thus from the preceding paragraph we have that $s_{Bl} \leq s_R$, $s_{\neg Bl} \leq s_{R\wedge Al}$, and $s_R \leq s_{R\wedge Al}$. However we can glean more information from these sentences. The semantic theory of N specifies that in the least exceptional, or simplest, worlds in which R is true, $R \supset Bl$ must also be true. (That is, in the simplest worlds in which there are ravens, these ravens are black.) Also in the simplest worlds in which $R \wedge Al$ is true, $R \wedge Al \supset \neg Bl$ must also be true. Thus, $R \supset Bl$ is true at every world in s_R, and $R \wedge Al \supset \neg Bl$ is true at every world in $s_{R\wedge Al}$. But this requires that s_R and $s_{R\wedge Al}$ are disjoint, and so $s_R < s_{R\wedge Al}$.

For reasoning by default, if all that is known is R, and we assume that the world being modelled, w, is as simple as possible, then it must be that $w \in s_R$. Since $R \supset Bl$ is true at every world of s_R we can conclude that Bl is true also. If on the other hand we know that $R \wedge Al$ is true at w, then assuming that the world is as simple as possible lets us conclude that $w \in s_{R\wedge Al}$; since $R \wedge Al \supset \neg Bl$ is true at all worlds in $s_{R\wedge Al}$ we would conclude $\neg Bl$.

Informally, the information implicit in a set of defaults and necessary statements **N** imposes constraints on any model of this set of statements. The idea is to first make this information explicit in a structure, called the *canonical structure*. The canonical structure $St(\mathbf{N})$ can be regarded as a "proto-model", that is, as a structure that in some sense represents the set of models of **N**, yet isn't itself a model. This structure consists of a set of "points", in some partial order. An individual point is denoted s_i, where i is some arbitrary index or, as we shall use for notation, s_A where A is some formula of classical logic. A point may be regarded as representing a set of mutually accessible worlds, and the partial order relation between points represents accessibility between two such sets. Sets of formulae are associated with each point; for example, $R \supset Bl$ must be true at s_R. The notion of truth is defined in this structure, and for conditional and necessary statements, is shown to correspond to that of validity in the logic N. Default reasoning is easily defined in this structure. Basically, for given contingent information **C** about a domain, we locate a set of points, S_W, corresponding to the sets of least exceptional worlds consistent with this information. This set of points has a direct correspondence with an *extension*, or reasonable set of beliefs that may be held by default. If a property follows in one of these extensions then it can be shown to follow in one of the *maximal contingent extensions* in the approach of [Del88]. If a property follows in all of these extensions then it follows by default in the approach of [Del88].

A key point is that the conditional operator $\Rightarrow$ does not appear in the canonical structure; thus, the complexity of default inferencing is that of reasoning at a point, together with the overhead of manipulating the canonical structure. In the case of an inheritance hierarchy, both the constituents of the conditionals and the world knowledge are given by conjunctions of possibly-negated primitive propositions; hence (it will prove to be the case that) inheritance reasoning is efficient.

4 The Approach: Formal Details

An *inheritance network* is a set **N** where if $\alpha \in \mathbf{N}$ then α is of the form $\Box(A \supset B)$, $A \Rightarrow B$, or $\neg(A \Rightarrow B)$.[3] Each antecedent and consequent in a conditional are conjunctions of possibly-negated primitive propositions. $\Box(A \supset B)$ is interpreted as "if A then necessarily B"; $A \Rightarrow B$ is interpreted as "if A then, in the normal course of events, B"; $\neg(A \Rightarrow B)$ denies a default without asserting a default conditional between the antecedent and the negation of the consequent. An example of an inheritance network is $\{US \Rightarrow A, \Box(A \supset P), A \Rightarrow E$, and $\neg(US \Rightarrow E)\}$. That is, informally, university students are normally adults; adults are necessarily persons; adults are normally employed; and it is not the case that university students are normally employed. The negated default $\neg(US \Rightarrow E)$ simply blocks the transitivity in the first and third conditionals. If we wanted to conclude that university students normally were not employed, we would use $US \Rightarrow \neg E$ in place of the last default.

In addition, we are given a set of possibly-negated primitive propositions **C** which represents known facts. **C** then represents the known contingent information that is true at the domain (or world) being modelled; **N** in contrast contains statements whose truth is determined by reference to other (less exceptional) possible worlds. The goal is to specify precisely what follows by default from **C** and **N**. The approach then is somewhat more general than most other approaches to inheritance hierarchies: we allow mixed strict and default conditionals; we allow negated defaults; and we allow negated primitive propositions.[4]

The general approach is as follows. First we construct the canonical structure $St(\mathbf{N})$. Truth is defined for statements of the form $A \Rightarrow B$, $\neg(A \Rightarrow B)$ or $\Box(A \supset B)$ in this structure and is shown to correspond to the definition of truth in all models of N (and so validity in N). Default inferencing in $St(\mathbf{N})$, appropriately defined, corresponds to the approach of [Del88]. Thus there are three steps in the process:

1. Construct the canonical structure $St(\mathbf{N})$ from **N**.
2. Determine a set of "relevant" points in $St(\mathbf{N})$, according to **C**. These points yield the least exceptional worlds which may be the case, given **C** and assuming that the world being modelled is as unexceptional as possible.
3. Determine default properties by means of (non-default) reasoning at these points.

4.1 The Canonical Structure

The *canonical structure* $St(\mathbf{N})$ of **N** is the poset $[S; \leq]$ resulting from the construction given below. The elements s_i of S are called *points* of the canonical structure. Each $s_i \in S$ is an ordered pair $< s_i(\exists), s_i(\forall) >$ of sets of formulae. If points are interpreted as standing for sets of

[3] We could allow negations of necessary entailments but they would have no effect on the algorithm.

[4] However, as will be seen, we do not allow unrestricted reasoning with negation.

mutually accessible worlds, then $s_i(\forall)$ is the set of formulae that must be true at every such world; $s_i(\exists)$ is a set of formulae where each formula must (individually) be true at some world in s_i. Consider for example where s_i is

$$< \{A, B\}, \{A \supset C\} > .$$

This point represents some set of possible worlds, where the only constraints on this set of worlds are, first, that there is some world where A is true and and another (possibly the same world) where B is true and, second, that $A \supset C$ is true at every world in this set.

Since we will frequently be talking about the least set of worlds (or point) that contains a world in which some formula is true, it will prove convenient to index points by formulae also. The notation s_A then is intended to stand for the least set of worlds (represented by a point) in which there is a world where A is true. We can do this formally as follows.

Definition 4.1 *s_A is the element $s_i \in S$ (if it exists) where for some B, $A \equiv B$ and $B \in s_i(\exists)$.*

That is, if $B \in s_i(\exists)$ then s_i represents the set of worlds in which there is a world where B is true and, furthermore, B is not true at any world known to be simpler (or strictly less exceptional) than those in s_i. If $A \equiv B$ then these same considerations apply to A. Thus s_A represents the set of worlds containing the least world in which A is true.

In the construction given below we will on occasion assert that $s_i \leq s_j$: that the set of worlds represented by i is no more exceptional than the set represented by j. If we discover that both $s_i \leq s_j$ and $s_j \leq s_i$ then this means that s_i and s_j represent the same set of worlds. We write this last result as $s_i = s_j$. In this case, we can *merge* the points s_i and s_j into a single point. That is, the points s_i and s_j are replaced by the single point $< s_i(\exists) \cup s_j(\exists), s_i(\forall) \cup s_j(\forall) >$. Thus for example if we have the (rather cumbersome) assertion that birds are normally flying birds, $B \Rightarrow B \wedge F$, then it proves to be the case that $s_B = s_{B \wedge F}$.[5] Thus the least set of worlds in which there are birds is the same as the least set in which there are flying birds. Note that this means that for some point s_i we would have that both $B \in s_i(\exists)$ and $B \wedge F \in s_i(\exists)$. I will use the notation $s_i \cup s_j$ to denote the result of merging s_i and s_j.

For the canonical structure, I begin by defining the set of primitive propositions known to be true at some world at a point:

Definition 4.2 *If A is a conjunction of (possibly-negated) primitive propositions, then $Atoms(A)$ is the set of (possibly-negated) primitive propositions in A.*

Thus $Atoms(A \wedge B \wedge \neg C)$ is $\{A, B, \neg C\}$.

Definition 4.3 *$Cl(s_i)$ is defined by:*

1. *if $A \in s_i(\exists)$ then if $p \in Atoms(A)$ then $p \in Cl(s_i)$;*
2. *if $A \supset B \in s_i(\forall)$ and $Atoms(A) \subseteq Cl(s_i)$ then if $p \in Atoms(B)$ then $p \in Cl(s_i)$.*

[5]Since $B \wedge F \Rightarrow B$ is a theorem of N, we obtain that $B \wedge F \Leftrightarrow B$ in this example.

$Cl(s_i)$ is the set (mnemonically, *closure*) of primitive propositions known to be true at some world in s_i. Thus if s_i is $< \{A \wedge \neg B, C\}, \{A \supset P, C \supset Q\} >$ then $Cl(s_i)$ is $\{A, \neg B, C, P, Q\}$. It proves to be the case in the construction below that if **N** is consistent then if $A \in Cl(s_i)$ then $\neg A \notin Cl(s_i)$

Construction 4.1 *1. Initialise the structure by:*

- *(a) There is one designated point s_w where s_w is $< \{\bigwedge(p_i)$ for all $p_i \in \mathbf{C}\}, \emptyset >$.*
- *(b) For $(A \Rightarrow B) \in \mathbf{N}$, $s_i \longleftarrow < \{A\}, \{A \supset B\} >$ for distinct s_i.*
- *(c) For $\neg(A \Rightarrow B) \in \mathbf{N}$, $s_i \longleftarrow < \{A, A \wedge \neg B\}, \emptyset >$ for distinct s_i.*
- *(d) For $\Box(A \supset B) \in \mathbf{N}$, $s_i(\forall) \leftarrow s_i(\forall) \cup \{A \supset B\}$ for every $s_i \in S$.*

2. If $A \in s_i(\exists)$, $B \in s_j(\exists)$ and $B \in Cl(s_i)$ then assert $s_j \leq s_i$.

3. If $s_i \leq s_j$ and there is $A \in Cl(s_i)$ and $\neg A \in Cl(s_j)$ then assert $s_i < s_j$.

The point s_w represents the set of worlds containing the world corresponding to the domain being modelled. Step 1b specifies that in the set of simplest worlds in which A is true at some world, $A \supset B$ is true at every world in the set. Step 1c represents negated defaults in the canonical structure: if $\neg(A \Rightarrow B)$ is true, then at the simplest worlds in which A is true, it is *not* the case that B is necessarily true; hence at one of the simplest worlds in which A is true $\neg B$ is also true. The point $< \{A, A \wedge \neg B\}, \emptyset >$ then contains the information that the set of simplest worlds containing a world where A is true is the same as the set of simplest worlds containing a world where $A \wedge \neg B$ is true. This can also be expressed as $s_A = s_{A \wedge \neg B}$. Step 2 is used primarily to assert $s_B \leq s_A$ when $A \Rightarrow B$ is a default and B occurs as the antecedent of another default. That is, if B is true at some world in s_i then if s_j contains the least world in which B is true then $s_j \leq s_i$ Note that if $A \in s_i(\exists)$ and $B \in s_j(\exists)$, and $A \in Cl(s_j)$ and $B \in Cl(s_i)$ then we obtain $s_i = s_j$ and so merge s_i and s_j. This step also serves to locate s_w in the structure, and so serves to account for the assumption that the world modelled by **C**, w, is as unexceptional as possible. For example, if we have that **N** is $\{R \Rightarrow Bl, R \wedge Al \Rightarrow \neg Bl\}$ and **C** is $\{R, Al\}$ then in the construction we obtain that $s_w = s_{R \wedge Al}$. Step 3 deals with the case where s_i and s_j are known to be non-equal.

The notion of truth of conditionals ($\models'$) can be defined in this structure and shown to correspond to truth in the logic N ($\models$):

Definition 4.4 *$St(\mathbf{N}) \models' A \supset B$ iff*

1. *$s_A(\forall) \models A \supset B$ if s_A exists,*
2. *$s_w(\forall) \models A \supset B$ otherwise.*

Theorem 4.1 *If $St(\mathbf{N}) \models' A \supset B$ then $\mathbf{N} \models A \Rightarrow B$.*[6]

[6]In [Del90] the theorem is generalised to an "if and only if".

4.2 Default Reasoning

Given $St(\mathrm{N})$, the next step is to define default reasoning with respect to this structure. The general approach is as follows. The canonical structure provides a number of points, representing sets of possible worlds, which contain sets of formulae that must be true at one, or at all, worlds represented by a point. The ordering $\leq$ between points corresponds to the "less exceptional than" accessibility relation. We also have some world knowledge **C**; the world modelled by **C**, w, is located somewhere in this structure. This locating of s_w corresponds to the assumption that w is as unexceptional as possible. Default reasoning then, as a first approximation, corresponds to what may be derived at the point s_w. Thus if N is $\{R \Rightarrow Bl, R \wedge Al \Rightarrow \neg Bl\}$ and' **C** is $\{R, Al\}$ then we obtain that $s_w = s_{R\wedge Al}$. Since $R \wedge Al \supset \neg Bl$ at every world in $s_{R\wedge Al}$ it follows that $\neg Bl$ must (by default) be true.

There is however a bit more to it than this. For example we may have that $s_A \leq s_w$ and $s_B \leq s_w$. Since we want to make the maximal set of "reasonable" default inferences, I assume that equality holds between s_w and these other points (whenever consistent). Note that it is only by making such an assumption that we can conclude from A and $A \Rightarrow B$ and $B \Rightarrow C$ that C follows by default. Hence given $s_A \leq s_w$ and $s_B \leq s_w$ we would want to assume, if we could, that $s_A = s_B = s_w$. However it could be that s_A together with s_w is consistent and that s_B together with s_w is consistent, but that the three points taken together are inconsistent. Thus we must allow for the possibility that there is more than one such maximal set. This leads to two possibilities with respect to default inferencing: if something follows by default in one such maximal set, then we have a notion of "credulous" default inference; for something that follows by default in all such sets, we have a much more conservative notion of default inference. These possibilities are covered below; the next section provides some examples.

Given the canonical structure $St(\mathrm{N})$ and contingent information **C**, default inferencing is defined as follows. We define a set of points S_W that gives the maximal set of points that the domain being modelled may be a member of. As mentioned, this "extension" of s_w, represented by S_W accounts for default transitivities (that is, transitivities among defaults such as $A \Rightarrow B$ and $B \Rightarrow C$).

Definition 4.5 *$S_W \subseteq S$ is a maximal set of points in $St(\mathrm{N})$, such that:*

1. *$s_w \in S_W$;*
2. *if $s_i \in S_W$ then $s_i \leq s_w$;*
3. *if $s_i \in S_W$ and $s_i \leq s_j \leq s_w$ then $s_j \in S_W$;*
4. *S_W is consistent (i.e there is no A such that $A, \neg A \in Cl(\bigcup\{s_i | s_i \in S_W\})$).*

Recall that the operator $\cup$ applied to points s_i and s_j signifies the merging of the points, $< s_i(\exists) \cup s_j(\exists), s_i(\forall) \cup s_j(\forall) >$.

Definition 4.6 *For S_W defined above,*

1. *A follows as a* credulous default inference *if $Atoms(A) \subseteq Cl(\bigcup\{s_i | s_i \in S_W\})$.*
2. *A follows as a* (general) default inference *if $Atoms(A) \subseteq Cl(\bigcup\{s_i | s_i \in S_W\})$ for every such S_W.*

We also obtain:

Theorem 4.2 *1. If A follows as a credulous default inference then A follows by default in some maximal contingent extension in the approach of [Del88].*

2. If A follows as a general default according to the above conditions, then A follows by default in the approach of [Del88].

Thus the examples of default inferences in [Del88] provide examples for the present approach. The following section discusses additional examples.

4.3 Complexity Considerations

The procedure consists of two parts: constructing the canonical model and reasoning within the structure. Both parts however are essentially the same with respect to complexity considerations. In both cases the complexity hinges on two facts: first that there are no more points in $St(\mathrm{N})$ than there are elements of N, and second that the complexity of reasoning at a point is determined by the form of the formulae in N where the $\Rightarrow$ operator is replaced by $\supset$.

To begin with, constructing or searching the canonical structure is $O(n^2)$, where $n = | \mathrm{N} |$. This is because the construction of $St(\mathrm{N})$ depends only on the set of conditionals in N: the number of points is $O(n)$ and the number of connections between points is $O(n^2)$ in the worst case. (Note too that while in the worst case manipulating the structure is $O(n^2)$, in practice it appears to be roughly linear.)

The additional overhead of reasoning with elements of N in the construction of the canonical structure is $O(m)$ where m is the total length of the formulae in N. That is, the antecedents and consequents of the elements of N are conjunctions of primitive propositions. In the construction, $\Rightarrow$ in these formulae is replaced with $\supset$. By the definition of Cl and the construction of the canonical structure, we effectively ignore negated primitive propositions and only "apply" defaults in a "forward" direction. Hence the complexity of reasoning with formulae at a point is proportional to the total length of formulae at a point, m [Ull82], and so the whole procedure is $O(n^2 m)$. Alternatively the issue of reasoning with formulae at a point is equivalent to the satisfiability of propositional Horn clauses [DG84], since we ignore negation, and conjunctions in the consequences have no effect on the algorithm.

5 Examples

Consider first the defaults $P \Rightarrow Q$ and $Q \Rightarrow R$. We obtain that s_P is $< \{P\}, \{P \supset Q\} >$ and s_Q is $< \{Q\}, \{Q \supset R\} >$. In addition, from step 2 of the construction, $s_Q \leq s_P$. If in addition we have that $\mathbf{C} = \{P\}$ then $s_w = s_P$ (again by step 2 of the construction). S_W is $\{s_P, s_Q\}$; $Cl(s_P \cup s_Q)$ is $\{P, Q, R\}$; and so R follows as both a credulous and general default inference.

For a second example, consider the following assertions:

1. Quakers are normally pacifists;
2. republicans are normally not pacifists;
3. pacifists are normally anti-military;
4. non-pacifists are normally not anti-military;
5. Quakers normally have strongly-held beliefs;
6. republicans normally have strongly-held beliefs;
7. RN is a Quaker;
8. RN is a republican.

Hence $\mathbf{N} = \{Q \Rightarrow P, R \Rightarrow \neg P, P \Rightarrow AM, \neg P \Rightarrow \neg AM, Q \Rightarrow SHB, R \Rightarrow SHB, \Box(RN \supset Q), \Box(RN \supset R)\}$. From steps 1a-1c of the construction $St(\mathbf{N})$ has the points:

s_Q *is* $< \{Q\}, \{Q \supset P, Q \supset SHB\} >$
s_R *is* $< \{R\}, \{R \supset \neg P, R \supset SHB\} >$
s_P *is* $< \{P\}, \{P \supset AM\} >$
$s_{\neg P}$ *is* $< \{\neg P\}, \{\neg P \supset \neg AM\} >$.

In addition we obtain (step 1d of the construction) that the formulae $RN \supset Q$ and $RN \supset R$ are true at every point. From step 2 we obtain that $s_P \leq s_Q$ and $s_{\neg P} \leq s_R$.

If we have that $\mathbf{C} = \{RN\}$ then s_w is $< \{RN\}, \{RN \supset Q, RN \supset R\} >$ and from step 2 we also obtain that $s_Q \leq s_w$ and $s_R \leq s_w$. For default reasoning, we can assume that $s_w = s_Q$ or $s_w = s_R$. However, assuming $s_w = s_Q = s_R$ leads to an inconsistency, since both P and $\neg P$ follow in this case. Hence there are two extensions, corresponding to the case where Sw is $s_w \cup s_Q$ and Sw is $s_w \cup s_R$. In the first extension P, AM, and SHB are true and in the second $\neg P$, $\neg AM$, and SHB are true. In both extensions of course RN, Q, and R are true. If we intersect the extensions, we obtain the "firm" default conclusion that SHB is true. Note that if we had that $RN \Rightarrow Q$, $RN \Rightarrow R$ then we would obtain the same set of inferences except that Q and R would now follow as default conclusions in both extensions.

For a final example, consider where $\mathbf{N} = \{A \Rightarrow B, B \Rightarrow C, B \Rightarrow D, C \Rightarrow E, D \Rightarrow \neg E\}$. This example is the same as Figure 4 in [THT87]. From steps 1a-1c of the construction $St(\mathbf{N})$ has the points:

s_A *is* $< \{A\}, \{A \supset B\} >$
s_B *is* $< \{B\}, \{B \supset C, B \supset D\} >$
s_C *is* $< \{C\}, \{C \supset E\} >$
s_D *is* $< \{D\}, \{D \supset \neg E\} >$.

Step 2 yields $s_C \leq s_B \leq s_A$ and $s_D \leq s_B \leq s_A$.

If we have that $\mathbf{C} = \{A\}$ then s_w is simply $< \{A\}, \emptyset >$ and from step 2 we obtain that $s_w = s_A$. For default reasoning, there are two extensions, one in which $\{A, B, C, D, E\}$ are true and another in which $\{A, B, C, D, \neg E\}$ are true. If we intersect the extensions, then we obtain the "firm" default conclusions $\{A, B, C, D\}$. If $\mathbf{C}$ was $\{B\}$ then we would obtain the same extensions, except of course A would no longer be concluded.

Consider now where $\mathbf{C} = \{A\}$ as before, and $\mathbf{N}$ has in addition $A \Rightarrow \neg D$ (Figure 5 in [THT87]). Thus the transitivity $A \Rightarrow B$, $B \Rightarrow D$ is explicitly blocked. We obtain now that

s_A *is* $< \{A\}, \{A \supset B, A \supset \neg D\} >$.

Crucially, from step 3 of the construction, we also obtain that $s_B < s_A$. That is, the simplest worlds where A is true must have both B and $\neg D$ true; the simplest worlds where B is true must also have D true; hence s_A and s_B must be disjoint. This means then that for default reasoning we obtain now only one extension, where $\{A, B, \neg D\}$ are true.

This last result is interesting in that the conclusion of C from $A \Rightarrow B$ and $B \Rightarrow C$ is also blocked. From the point of view of inheritance diagrams, such as those employed in [THT87], this seems quite unreasonable: there is after all a path from A to C with nothing obviously blocking it. However I would suggest that this example instead points out a limitation with inheritance diagrams as a means of justifying defeasible inheritance. The approach presented here provably rests on a logic of defaults together with an approach for defeasible reasoning. If we assume that the logic does indeed fairly formalise a reasonable approach to representing default properties for naturally occurring kinds, and that the approach for default reasoning does indeed capture a reasonable approach to default inference, then it necessarily follows that C not be a default conclusion in this case. Moreover the rationale for this falls out from the semantic theory: the simplest worlds in which A is true are distinct from the simplest worlds in which B is true. Thus we can conclude B by default from A but we cannot subsequently conclude C from B.[7] We cannot conclude C from B in this case since this would rely on the assumption, in our notation, that $s_A = s_B$; however in the above example we have $s_A \leq s_B$. This distinction is certainly not apparent in the corresponding inheritance diagram; in addition there seems to be no way this distinction *could* reasonably be made.

6 Discussion

This paper has presented an algorithm for inheritance reasoning based on the model theory of a logic of defaults. The overall approach is to begin with an existing semantic theory underlying an approach to default reasoning, and to derive the corresponding system of inheritance reasoning. A set of strict and defeasible conditional assertions is translated into a particular mathematical structure called the canonical structure. This structure in a precise sense represents the set of all models of these assertions. The structure then directly reflects intuitions explicitly or implicitly contained in the set of defaults. Defeasible inferencing is correct, in that it is shown to be sound with respect to the approach of [Del88]. Thus the semantic theory sanctions this approach to the inheritance of properties.

In the terminology of [THT87] the system is *credulous* in that initially we try to conclude as much as possible; something of a skeptical reasoner is obtained by considering conclusions that occur is all credulous extensions. The system is *downward* in that properties are seen as "flowing" from classes to their subclasses, unless blocked by an exception. Hence the system is also *coupled* in that

[7] Of course if $\mathbf{C} = \{B\}$ we could conclude both C and D; moreover in the full logic we could also conclude $\neg A$.

a subclass is always in agreement with a superclass, with respect to inheritance, unless explicitly blocked by an exception. In addition defeasible and strict links (negated or not) may be mixed arbitrarily. Thus this work provides a semantics for this class of systems. Finally, this approach arguably shows that inheritance diagrams may be simply too limited as a notational or heuristic device to adequately capture even simple inheritance networks.

It is shown that inheritance reasoning may be efficiently carried out in this structure. This is because, first, the size of the canonical structure is bounded by the size of **N**, and second because the complexity of reasoning at a point is determined by the form of the formulae in **N** where the $\Rightarrow$ operator is replaced by $\supset$. The complexity intrinsic in constructing and manipulating the canonical structure itself is $O(n^2)$, where $n = |\mathbf{N}|$. Since the formulae that we deal with are restricted to be conditionals involving conjunctions of primitive propositions, and since conditionals are "applied" in a forward direction only, the complexity of reasoning with a set of formulae at a point is proportional to the total length of all formulae, m. Hence the overall complexity is $O(n^2m)$.

Acknowledgements

I would like to thank the referees for their helpful and constructive comments on this paper. This research was supported by the Natural Science and Engineering Research Council of Canada grant A0884.

References

[Bou89] C. Boutilier. A semantical approach to stable inheritance reasoning. In *IJCAI-89*, pages 1134–1139, 1989.

[Del87] J.P. Delgrande. A first-order conditional logic for prototypical properties. *Artificial Intelligence*, 33(1):105–130, 1987.

[Del88] J.P. Delgrande. An approach to default reasoning based on a first-order conditional logic: Revised report. *Artificial Intelligence*, 36(1):63–90, 1988.

[Del90] J.P. Delgrande. A (sometimes) efficient, provably correct algorithm for default inferencing. Technical report, School of Computing Science, Simon Fraser University, 1990. In preparation.

[DG84] W.F. Dowling and J.H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Logic Programming*, 1(3):267–284, 1984.

[ER83] D.W. Etherington and R. Reiter. On inheritance hierarchies with exceptions. In *Proc. AAAI-83*, pages 104–108, 1983.

[Gef88] H. Geffner. On the logic of defaults. In *Proc. AAAI-88*, St. Paul, Minnesota, 1988.

[Hau88] B.A. Haugh. Tractable thoeries of multiple defeasible inheritance in ordinary nonmonotonic logics. In *Proc. AAAI-88*, St. Paul, 1988.

[HT88] J.F. Horty and R.H. Thomason. Mixing strict and defeasible inheritance. In *AAAI-88*, pages 427–432, 1988.

[HTT87] J.F. Horty, R.H. Thomason, and D.S. Touretzky. A skeptical theory of inheritance in nonmonotonic semantic networks. In *AAAI-87*, 1987.

[MD80] D.V. McDermott and J. Doyle. Nonmonotonic logic I. *Artificial Intelligence*, 13:41–72, 1980.

[Moo83] R.C. Moore. Semantical considerations on nonmonotonic logic. In *Proc. IJCAI-83*, pages 272–279, Karlsruhe, West Germany, 1983.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Sho88] Y. Shoham. *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence*. The MIT Press, Cambridge, Mass., 1988.

[THT87] D.S. Touretzky, J.F. Horty, and R.H. Thomason. A clash of intuitions: The current state of nonmonotonic multiple inheritance systems. In *Proc. IJCAI-87*, pages 476–482, Milan, 1987.

[Ull82] J.D. Ullman. *Principles of Database Systems*. Computer Science Press, Inc., Rockville, Maryland, 1982.

A Partial Semantics for Nonmonotonic Logics

Wayne Wobcke
Knowledge Systems Group
Basser Dept. of Computer Science
University of Sydney, Sydney 2006
AUSTRALIA

Abstract

One problem with existing methods of formalizing nonmonotonic reasoning such as Non-Monotonic Logics I and II, Default Logic and Autoepistemic Logic, is that apparently a different notion of consistency is being employed in each of the formalisms. In an attempt to provide a common semantic basis for these formalisms, we develop a logic of consistency and use it to reconstruct each of the above nonmonotonic logics. We are successful in providing a common underlying partial semantics for the nonmonotonic systems, but our result indicates that the use of consistency plays only a minor part in these systems' behaviour.

1 Introduction

Many attempts to formalize nonmonotonic reasoning are based on inferring nonmonotonic conclusions if they are 'consistent' with what is known or believed. McDermott and Doyle's [1980] Non-Monotonic Logic I and McDermott's [1982] Nonmonotonic Logic II include formulae of the form Mϕ, which they explicitly recommend be read as 'ϕ is consistent'. A theory in such a language is intended to encode all the information about which formulae are consistent with it. Reiter's [1980] Default Logic uses the notation Mϕ in a meta-logical way as a shorthand for ϕ being consistent with a first-order theory currently under consideration. Moore's [1985] Autoepistemic Logic, although based on the beliefs of an ideally rational agent, can be formulated as being based on a notion of consistency – an agent believes ϕ iff $\sim\phi$ is not consistent with its beliefs.

One problem with these methods of formalizing nonmonotonic reasoning is that on the face of it, a different notion of consistency is being employed in each of the formalisms. Thus the question arises as to whether the notions really are distinct or whether they are variations of the same underlying notion of consistency. This motivates the work described in this paper, in which we present a logic of consistency and use it to reconstruct each of the above nonmonotonic logics. Our semantics is based on our possible-worlds version of situation semantics, [Wobcke, 1988], and in particular, on a hierarchy of situations (partial states of affairs). We prove soundness and completeness results for our logic. We are successful only to a limited extent in providing a common underlying partial semantics for the nonmonotonic systems, because the use of consistency plays only a minor part in the systems' behaviour. In particular, other aspects such as the construction of extensions in Default Logic and the groundedness of beliefs in Autoepistemic Logic, are more important in sanctioning inferences. We conclude, therefore, that nonmonotonic logics based purely on consistency are too weak to warrant drawing even the most intuitive of nonmonotonic conclusions.

2 The Consistency Logic SE

In this section, we define the syntax and semantics of our logic of consistency. The basic formulae of our logic are of the form $c:\phi$, where c denotes an information state and ϕ a fact; such a formula means that ϕ holds in the state $[c]$. Information states are formally reconstructed as sets of possible worlds: ϕ holds in a state σ if ϕ holds in all worlds in the set. The formulae ϕ denoting facts can be constructed from the propositional calculus connectives and the modal operator 'M'. The formula $c:$ Mϕ may be interpreted as 'ϕ is consistent with what facts hold in the state $[c]$'. To formally define the truth conditions for our logic, we adapt the standard modal logic semantics for possibility: Mϕ holds at a state σ if ϕ holds in some information state accessible to σ. We will define matters such that the states accessible to an information state satisfy just those atomic facts that are consistent (with respect to the logic) with the atomic facts holding at σ. Such an accessible state we call an *extension* of the original state. We now define a logic for consistent extensions which we call SE, for *situated extensions*.

2.1 Situated Extensions: Syntax

In stating some of the axioms for SE, we need to distinguish modal from non-modal formulae. One reason for this is that disjunction distribution fails for atomic formulae ($\phi \vee \psi$ can hold in an information state without either disjunct in particular holding), but if ϕ (say) is an atomic modal formula, it must be determinately true or false, hence if ϕ is false, the only way for the disjunction to hold is for ψ to hold: thus a limited form of disjunction distribution holds in SE.

Definition. A *modal literal* is a formula of the form Mϕ or $\sim$Mϕ, where ϕ is a proposition.

Definition. A *literal* is a PC formula or a modal literal.

The following scheme defines the axioms of the logic SE:

(P1) all the PC axioms written with formulae $c : \phi$,
(P2) $c : \phi$, for all c and all PC axioms ϕ,
(P3) $c : (\phi \rightarrow \psi) \rightarrow (c : \phi \rightarrow c : \psi)$,
(P4) $c : \sim\phi \rightarrow \sim c : \phi$,
(E1) $c : \mathsf{M}(\phi \rightarrow \psi) \rightarrow (\mathsf{M}\phi \rightarrow \mathsf{M}\psi)$,
(E2) $c : \mathsf{M}\mathsf{M}\phi \rightarrow \mathsf{M}\phi$,
(E3) $c : \mathsf{M}\sim\phi \leftrightarrow \mathsf{M}\sim\mathsf{M}\phi$,
(E4) $c : \sim\phi \rightarrow c : \sim\mathsf{M}\phi$, for ϕ a non-modal literal,
(E5) $\sim c : \phi \rightarrow c : \mathsf{M}\sim\phi$,
(E6) $c : \mathsf{M}(\phi \vee \psi) \rightarrow (\mathsf{M}\phi \vee \mathsf{M}\psi)$, for ϕ a modal literal,
(E7) $c : \sim\mathsf{M}(\phi \,\&\, \mathsf{M}\sim\phi)$.

The only inference rule in SE is modus ponens, i.e.

(MP) From α and $\alpha \rightarrow \beta$ infer β.

Axiom schema (P1) (which generates formulae like $(c : p \,\&\, c : q) \rightarrow c : q$) encodes the fact that the propositional connectives apply to facts holding in information states in the same way as for classical propositions. Axiom schema (P3) ensures that the set of facts holding in each state is deductively closed. Axiom schema (P2) in conjunction with (P3) ensures that every PC theorem is true in every state. Axiom schema (P4) captures a notion of consistency of states: if $\sim\phi$ holds in some state, then ϕ doesn't hold.

It will be useful in what follows to refer to sets of formulae that can hold at a single information state. These are SE formulae stripped of their context symbols:

Definition. Let E be that system consisting of the PC axioms and (MP), plus (E1) to (E7) (without c's), but without (E4) and (E5).

2.2 Situated Extensions: Semantics

SE interpretations (like standard S4 interpretations) consist of a set of information states together with a reflexive and transitive accessibility relation on that set. The interpretation of a context symbol c is a state, which is a set of possible worlds. A formula $c : \phi$ where ϕ is an *atomic* proposition (i.e. contains no modal operator) is true just if ϕ holds in all worlds in the set $[c]$ (we write $[c] \models \phi$). This truth condition is identical to van Fraassen's [1966] supervaluation semantics. Now each SE interpretation specifies a hierarchy of states, with $\mathsf{M}\phi$ holding at σ just if ϕ holds at some state accessible to σ. We further stipulate that the set of atomic facts holding at any state accessible to a given state σ must contain the set of atomic facts holding at σ – one way to do this is to make the hierarchy of information states correspond to the set inclusion ordering on sets of possible worlds (i.e. the more worlds in the set, the fewer facts hold in the state). Thus the set of facts holding at an information state σ completely determines the hierarchy of information states below it.

Definition. Given a set of *atomic* E sentences Γ, the *hierarchy of consistent extensions* for Γ is the hierarchy of information states each of which corresponds to a set of atomic sentences containing Γ, with the accessibility relation defined on this hierarchy by inclusion on the sets of worlds comprising each state.

Formally, the truth conditions for SE formulae are defined as follows. Let I be an SE interpretation. Then we define the truth conditions for formulae with respect to I in two stages; first for the atomic SE formulae, then for the complex SE formulae. Truth conditions for modal formulae are defined only for formulae in conjunctive normal form (the truth value of a non-CNF formula is defined as the truth value of the unique equivalent CNF formula). Let $[c]$ be the information state that is the interpretation of c and consider the hierarchy of states I_c rooted at $[c]$. Then, for the atomic SE formulae $c : \phi$:

$I_c \models \phi$	if $[c] \models \phi$, for all atomic propositions ϕ,
$I_c \models \sim\phi$	if $I_c \not\models \phi$, for ϕ non-atomic,
$I_c \models \phi \vee \psi$	if $I_c \models \phi$ or $I_c \models \psi$, for ϕ or ψ non-atomic,
$I_c \models \phi \,\&\, \psi$	if $I_c \models \phi$ and $I_c \models \psi$, for ϕ or ψ non-atomic,
$I_c \models \mathsf{M}\phi$	if some state accessible to $[c]$ satisfies ϕ.

The truth conditions for complex SE formulae are as follows:

$I \models c : \phi$	if $I_c \models \phi$,
$I \models \sim c : \phi$	if $I_c \not\models \phi$,
$I \models c : \phi \vee c' : \psi$	if $I_c \models \phi$ or $I_{c'} \models \psi$,
$I \models c : \phi \,\&\, c' : \psi$	if $I_c \models \phi$ and $I_{c'} \models \psi$.

Proposition. SE is a sound and complete inference system.

Proof. (Sketch) Soundness is relatively straightforward. Completeness is proven using the standard Henkin method to construct a model for any maximal consistent set of SE sentences. But where, in the standard proof, maximal consistent sets of propositions are used to define possible worlds in the model, our information states are partial and correspond to sets of E sentences with the following properties:

Definition. A set Γ_c of E formulae is *coherent* if whenever Γ_c contains $\sim\phi$ for ϕ a non-modal literal, Γ_c contains $\sim\mathsf{M}\phi$.

Definition. A set Γ_c of E formulae is *complete* if whenever Γ_c does not contain ϕ, Γ_c contains $\mathsf{M}\sim\phi$.

A coherent set of sentences that is consistent cannot contain both a non-modal literal $\sim\phi$ and the statement that ϕ is consistent with it. A complete set of E sentences contains full information about which sentences are consistent with it.

Lemma. (Correspondence Lemma) Deductively closed, consistent, coherent and complete sets of E formulae containing all the E axioms are in one-one correspondence with the hierarchies of consistent extensions.

The proof of the lemma is the heart of the completeness proof: it involves constructing, in the manner of [Hughes and Cresswell, 1968], a hierarchy of information states corresponding to such a set of sentences. The key fact is the following:

Lemma. If Γ_c containing $\mathsf{M}\phi$ is a deductively closed, consistent, coherent and complete set of E formulae, then there is a deductively closed, consistent, coherent and complete extension of Γ_c containing ϕ.

Note that there is an inductive element in constructing the desired extension because ϕ may itself be a modal formula.

It is then not difficult to show that the hierarchy of information states so constructed satisfies exactly those formulae in the maximal consistent set of SE sentences.

The full proof may be found in the extended version of this paper, [Wobcke, 1989]. □

What justifies our thinking of M as consistency with the facts holding at $[c]$ is the following consequence of the second lemma above, which relates the M operator to consistency in E:

Corollary. A deductively closed, consistent, coherent and complete set Σ of E formulae containing all the E axioms contains the formula $\mathsf{M}\phi$ iff ϕ is consistent with Σ.

3 A Look at Some Nonmonotonic Logics

The motivation for developing our logic of consistency is to enable us to define a formal semantics for nonmonotonic logics where our M operator can be interpreted as consistency with the facts. In this section, we consider in more detail these nonmonotonic logics, and examine the extent to which it is possible to do the reconstruction (restricting ourselves to the propositional case only). We consider, in turn, Nonmonotonic Logics I and II, Default Logic and Autoepistemic Logic, including a short summary of each. Reiter [1987] and Bell [1989] provide more complete surveys.

3.1 Non-Monotonic Logic I

McDermott and Doyle [1980] introduce Non-Monotonic Logic I (NMLI). The language of NMLI includes a sentential operator M (which should not be confused with our M) and allows sentences to be formed using all the PC connectives. An NMLI theory is a set of NMLI sentences. So, for example, the following is an NMLI theory:

$\{noon\ \&\ \mathsf{M}\, sun\text{-}shining \rightarrow sun\text{-}shining,\ noon,$
$\ eclipse \rightarrow \sim sun\text{-}shining\}$

From this theory, it is possible to prove *sun–shining*.

Provability in NMLI is defined in terms of fixed points. Let Th(A) be the deductive closure in PC of a set of NMLI formulae A. Define an operator NM_A on sets of NMLI formulae by:

$NM_A(S) = \mathrm{Th}(A \cup \mathrm{As}_A(S))$,

where As_A is the set of assumptions from S, given by

$\mathrm{As}_A(S) = \{\mathsf{M}\phi \mid \sim\phi \notin S\} - \mathrm{Th}(A)$

A set S is a *fixed point* of NM_A if $NM_A(S) = S$. McDermott and Doyle then define provability, denoted $\vdash$, from a base set A as follows: $A \vdash \phi$ iff ϕ is contained in all fixed points of NM_A.

NMLI models of theories A are pairs $<V, S>$ where V is a PC model of the atomic sentences in A, and S is a fixed point of NM_A. McDermott and Doyle give a very sketchy 'proof' that NMLI is sound and complete, i.e. $A \vdash \phi$ iff ϕ holds in all models of A. The definition of 'model' is however, suspect, as noted by Davis [1980], in particular in defining the truth condition for $\mathsf{M}\phi$. If, however, we take it that a model satisfies $\mathsf{M}\phi$ if $\mathsf{M}\phi \in S$, then completeness fails, as the following example shows. Take $A = \{\sim p,\ q \vee \mathsf{M}p\}$, where p and q are atomic. Now $\mathsf{M}p$ cannot be in any $\mathrm{As}_A(S)$ because $\sim p \in A$. So $\mathsf{M}p$ cannot be in a fixed point S of NM_A because otherwise there would be a proof in PC of $\mathsf{M}p$ from $A \cup \mathrm{As}_A(S)$: this is impossible because $\sim\mathsf{M}p$ must be consistent with $\mathrm{As}_A(S)$ and hence must be consistent in PC with $A \cup \mathrm{As}_A(S)$ since $\sim q \notin \mathrm{As}_A(S)$. Hence every model of A is a model of q. But $A \nvdash q$, because q is not necessarily contained in all fixed points of NM_A: it won't be contained in the fixed point consisting of all the $\mathsf{M}\phi$ except for those ϕ that are consequences of p. The diagnosis of the problem here is easy: we are not forced to have $\sim\mathsf{M}\phi$ provable from A if $\mathsf{M}\phi$ is inconsistent with A. This provides some motivation for reconstructing NMLI in SE, where this property does hold.

Now we ask whether M can in fact be interpreted as consistency with what is believed. Put more precisely, the property being sought is that $A \vdash \mathsf{M}\phi$ iff ϕ is consistent with A. Unfortunately, both halves of this purported equivalence fail, as can be seen from the following examples. First, clearly $\mathsf{M}p$ is contained in all fixed points of $A = \{\sim p, \mathsf{M}p\}$, and as McDermott and Doyle note, such fixed points exist. But it is equally clear that p is not consistent with A. Second, take $A = \{\mathsf{M}q \rightarrow \sim p,\ \mathsf{M}p \rightarrow \sim q\}$, which has two fixed points, one containing $\{\mathsf{M}q, \sim p\}$ but not $\mathsf{M}p$ and one containing $\{\mathsf{M}p, \sim q\}$ but not $\mathsf{M}q$. Thus $\mathsf{M}p$ is not contained in all fixed points of NM_A, but $A \cup \{p\}$ is not inconsistent, because it has a fixed point containing $\{p, \mathsf{M}p, \sim q\}$. So McDermott and Doyle's logic, as they say, 'fails to capture a coherent notion of consistency' (p. 69). This is the primary motivation for reconstructing NMLI in SE.

To carry out this reconstruction, we need to consider just what McDermott and Doyle intend in stating that M should represent 'consistency with what is believed'. By their definition of of NM_A, any fixed point S contains only those sentences that follow from A and the assumption set of S. But it would seem that if sentences in S are to be counted as beliefs, we should require only a much weaker property: that belief sets be those sets of sentences that include all the consequences of their assumptions. We can make this precise with the following definition of an operator NM:

$NM(S) = \mathrm{Th}(S \cup \mathrm{As}(S))$,

where As(S) is, by analogy to the above, the set of assumptions from S, given by

$\mathrm{As}(S) = \{\mathsf{M}\phi \mid \sim\phi \notin S\}$

Belief sets will now be taken as the fixed points of NM, but in defining provability from A, we will restrict attention to those fixed points of NM that contain A.

To define a translation from NMLI to SE, the idea is to have each NMLI theory A as the set of formulae holding at an information state. If we can set up a correspondence between fixed points and SE models, then by the completeness of SE, the consequences of A will include those formulae $c:\phi$ such that ϕ holds at all fixed points. Indeed, with our revised definition of the fixed point operator and the following definition, we can show that models of SE(A) correspond to fixed points of NM:

Definition. Given an NMLI theory A, the SE theory corresponding to A is the set $\mathrm{SE}(A) = \{c:\phi \mid \phi \in A\}$.

Proposition. The set of formulae satisfied in a model of SE(A) is a fixed point of NM that contains A.

Proof. Let S be the set of formulae holding at $[c]$ in our SE model. Obviously S contains A. Now S is clearly deductively closed and is contained in $\mathrm{Th}(S \cup \mathrm{As}(S))$. To show the converse, since S contains (E5), S contains $\mathsf{M}\phi$ whenever S does not contain $\sim\phi$. Hence As(S) is contained in S and so $S \subseteq \mathrm{Th}(S \cup \mathrm{As}(S)) \subseteq S$. Hence $S = NM(S)$. □

Proposition. If S is a fixed point of NM that contains A and is coherent, then S is a set of formulae satisfied in some model of SE(A).

Proof. S is complete because if S does not contain ϕ, $\mathsf{M}\sim\phi \in S$ by the definition of NM. The result then follows from the correspondence lemma. □

Corollary. A formula is derivable in the SE theory corresponding to an NMLI theory A only if it is contained in all coherent and complete fixed points of NM that contain A.

Now that we have been able, to a certain extent, to reconstruct NMLI in SE, in which M can be interpreted as 'consistency with what facts hold', there remains the question of where this leaves us in relation to the original problems of nonmonotonic reasoning. Unfortunately, it seems we are in a worse position than before. Consider the NMLI theory consisting of the single sentence $\{\mathsf{M}\phi \rightarrow \phi\}$, similar to the 'sun-shining' example above. In the reconstructed formulation, it is now not possible to infer ϕ from this theory, because although there is a fixed point containing $\{\phi, \mathsf{M}\phi\}$ as before, there is also another fixed point containing $\{\sim\phi, \sim\mathsf{M}\phi\}$. One diagnosis of this problem is that under the revised definition of the fixed point operator, it seems that spurious beliefs ϕ can be introduced, beliefs whose support does not depend on making any nonmonotonic assumptions of the form $\mathsf{M}\psi$. But it is more serious than that, because what does follow from $c : \mathsf{M}\phi \rightarrow \phi$ according to the rules of SE is $c : \mathsf{M}\sim\phi \rightarrow \sim\phi$! So when M is interpreted as consistency, simple rules like this are not even sufficient to ensure a system will prefer ϕ over its negation. The appropriate conclusion to draw is that in order to properly handle nonmonotonic reasoning, it is insufficient to rely on mere consistency with beliefs as modelled in SE: much stronger conditions are needed.

3.2 Nonmonotonic Logic II

McDermott [1982] motivates his development of Nonmonotonic Logic II (NMLII) by noting that in NMLI, there are no axioms whatever that relate specifically to M. Accordingly, there are too many fixed points of NM_A, and hence by restricting the behaviour of M by including such axioms, some extraneous fixed points might be ruled out. McDermott considers using the axioms from the well-known modal logics T, S4 and S5, i.e. $\mathsf{L}\phi \rightarrow \phi$, $\mathsf{L}\phi \rightarrow \mathsf{LL}\phi$ and $\mathsf{M}\phi \rightarrow \mathsf{LM}\phi$, where L is the dual of M.

McDermott defines the provability operator $\vdash$ based on fixed points of the operator NM_A, as above. That is, $A \vdash \phi$ if ϕ is contained in all fixed points of the operator NM_A, where, repeating the definitions, we have:

$$NM_A(S) = \mathrm{Th}(A \cup \mathrm{As}_A(S)),$$

and

$$\mathrm{As}_A(S) = \{\mathsf{M}\phi \mid \sim\phi \notin S\} - \mathrm{Th}(A)$$

The difference between this and NMLI is that 'Th' now refers to deduction in a modal system (such as T, S4, S5) with all the instances of the relevant axiom schemata contained in all fixed points of NM_A. It is also important to realize that in defining provability, McDermott modifies the definition of the rule of necessitation in these modal systems S from 'if $S \vdash \phi$, infer $\mathsf{L}\phi$' to 'from ϕ, infer $\mathsf{L}\phi$'.

The semantics of NMLII is given in terms of noncommittal modal models. A *noncommittal* modal model of a theory A is a (T, S4, S5) model M of A such that $M \models \mathsf{M}\phi$ for all those ϕ such that $\sim\phi$ is not satisfied in some model of $A \cup acc(M, A)$, where $acc(M, A)$, the set of 'accidentally-holding' formulae of M, is the set of $\mathsf{M}\psi$ satisfied in M that are not satisfied in all models of A. So the noncommittal models are those that attempt to reflect the fact that ϕ is not ruled out as a belief (by satisfying $\mathsf{M}\phi$) whenever ϕ is not in fact ruled out as a belief (if $\sim\phi$ doesn't hold in all models). McDermott then proves that M is a model of a fixed point of NM_A iff M is a noncommittal model of A, from which it follows that $A \vdash \phi$ iff ϕ is true in all noncommittal models of A.

One point of note is that McDermott shows that the nonmonotonic system based on S5 'collapses' to monotonic S5 (with modified rule of necessitation). That is, with S5 as a basis, $A \vdash \phi$ iff $A \vdash \phi$, so no nonmonotonic conclusions can be drawn in nonmonotonic theories based on S5. Hence McDermott recommends basing nonmonotonic theories on S4 or T.

Can M be interpreted as consistency with what is believed in (any of) the revised nonmonotonic logics? That is, is it true that $A \vdash \mathsf{M}\phi$ iff ϕ is consistent with A? First note that the counterexample to the 'if' half in the NMLI case has been countered: $\{\sim p, \mathsf{M}p\}$ is now inconsistent (because $\mathsf{L}\sim p \rightarrow \sim\mathsf{M}p$ is a theorem of all the logics). But the counterexample can easily be modified as follows (the same one works for each of T, S4 and S5). Take $A = \{\mathsf{M}p, \mathsf{M}\sim p\}$. Clearly A is consistent, and $\mathsf{M}p$ holds in all fixed points of NM_A. But $A \cup \{p\}$ is inconsistent, because it is inconsistent in each of the modal systems (it contains $\mathsf{L}p$, hence $\sim\mathsf{M}\sim p$). So the 'if' half of the equivalence fails. Our counterexample above for the 'only if' half still works for each of the nonmonotonic systems based on T, S4 and S5. So NMLII, although an improvement on NMLI, still does not completely capture a notion of consistency.

Our reconstruction of NMLI in SE was motivated primarily in order to interpret M as consistency, but it also achieves the effect that motivated McDermott in designing NMLII, namely, by strengthening the logic of M, some spurious fixed points have been ruled out. As we indicated above, coherence and completeness are desirable properties of fixed points, which we obtain using SE's logic of M. However, as shown above, our reconstruction suffers a fate similar to that of nonmonotonic S5, in that the logic is too strong for a nonmonotonic theory $\{\mathsf{M}\phi \rightarrow \phi\}$ to decide between ϕ and $\sim\phi$. The conclusion, as above, is that conditions stronger than consistency with the facts are required for nonmonotonic reasoning.

3.3 Default Logic

Reiter [1980] presents Default Logic. The main difference between Default Logic and McDermott and Doyle's nonmonotonic logics is that in Default Logic, the operator M is not part of an underlying logical language. Rather, it assumes a meta-logical status, although it is still to be read as 'consistency with the facts'. A default theory $<D, W>$ consists of a set W of first-order formulae representing the known facts, and a set D of inference rules, called *defaults*, which we will write as $\alpha, M\beta_1, \cdots, M\beta_n \vdash w$. A default states roughly that some formula w can be inferred in the presence of α provided $\{\beta_1, \cdots, \beta_n\}$ is consistent with what is known. If a default rule is used to infer w, we will say that w is a default conclusion of the theory.

Of special interest are the extensions of a default theory. Intuitively, a (consistent) extension of a default theory $<D, W>$ is a set of first-order formulae that includes W and as many default conclusions as possible (retaining consistency) and which includes no extraneous formulae that are not either in W to begin with or derived from W using the defaults. More precisely:

Definition. An *extension* E of a default theory $<D, W>$ is a fixed point of the operator Γ, where for any set of first-order formulae S, $\Gamma(S)$ is the smallest deductively closed set containing W such that if $\alpha, M\beta_1, \cdots, M\beta_n \vdash w$ is a default rule, then if $\alpha \in \Gamma(S)$ and $\sim\beta_1, \cdots, \sim\beta_n \notin S, w \in \Gamma(S)$.

This definition suggests a way of characterizing extensions: as those sets that can be constructed by starting with W and repeatedly adding default conclusions until the addition of any more would result in an inconsistency. That is, Reiter shows that E is an extension of a default theory $<D, W>$ iff $E = \bigcup_{i=0}^{\infty} E_i$, where

$$E_0 = W,$$
$$E_{i+1} = \mathrm{Th}(E_i) \cup \{w : \alpha, M\beta_1, \cdots, M\beta_n \vdash w \in D, \\ \alpha \in E_i \text{ and } \sim\beta_1, \cdots, \sim\beta_n \notin E\}.$$

Note, however, that the definition is not constructive due to the occurrence of E in the definition of E_{i+1}.

Just as with McDermott and Doyle's nonmonotonic logics, a default theory can have multiple extensions. For example, the default theory $\{Mq \vdash \sim p, Mp \vdash \sim q\}$ has two extensions, one containing $\sim p$ and one containing $\sim q$, just as the corresponding nonmonotonic theory has two fixed points. Note, however, that in general, there is no exact correspondence between extensions and fixed points.

It is clear that, in a limited sense, M can be read as 'consistent with the facts' in that for the atomic case only, $M\phi$ could be said to hold of an extension E whenever $\sim\phi \notin E$, i.e. ϕ is consistent with E. But of course $M\phi$ can never be contained *in* an extension, so can never be a belief. This provides one motivation for reconstructing Default Logic in SE, because in this system, M is interpreted as consistency and we can have beliefs of the form $M\phi$.

But perhaps the primary motivation for reconstructing Default Logic in SE stems from the fact that Reiter provided no semantics for Default Logic, although this has since been rectified to some extent by Etherington [1987] and Konolige [1987]. The aim is to tighten the connection between M and consistency. Our reconstruction proceeds by using the defaults as inference rules in E (recall that E is the system of SE formulae without their context symbols). This generalizes Default Logic in that it allows the consequents w of defaults to be arbitrary E formulae. Our definition of an extension simply replaces Reiter's condition $\sim\beta \notin E$ by the condition $M\beta \in E$:

Definition. An *extension* E of an SE default theory is a fixed point of the operator Γ, where for any set of E formulae S, $\Gamma(S)$ is the smallest deductively closed set containing W such that if $\alpha \,\&\, M\beta \vdash \gamma$ is a default rule, then if $\alpha \in \Gamma(S)$ and $M\beta \in S$, then $\gamma \in \Gamma(S)$.

Then, by analogy with Reiter, we have:

Proposition. E is an extension of an SE default theory $<D, W>$ iff $E = \bigcup_{i=0}^{\infty} E_i$, where

$$E_0 = W,$$
$$E_{i+1} = \mathbf{Th}(E_i \cup \{\gamma\text{: } \alpha \,\&\, M\beta \vdash \gamma \in D,\ \alpha \in E_i,\ M\beta \in E\}).$$

where $\mathbf{Th}$ of a set S of E formulae is the set Σ_c (Σ stripped of its context symbols) where $\Sigma = \mathrm{Th}_{\mathrm{SE}}(c : S)$, where $c : S$ is the set $\{c : \psi \mid \psi \in S\}$. That is, $\mathbf{Th}(S)$ is calculated by adding the context symbol c in front of all formulae in S, taking the SE deductive closure, then stripping the context symbols off again.

Proof. Following Reiter. First, let $E^* = \bigcup_{i=0}^{\infty} E_i$. Then E^* contains W, is deductively closed and if $\alpha \,\&\, M\beta \vdash \gamma$ is a default rule with $\alpha \in E^*$ and $M\beta \in E$, then $\gamma \in E^*$. Hence $\Gamma(E) \subseteq E^*$.

We first show that an extension $E = E^*$. Since $E = \Gamma(E)$, we have $E \subseteq E^*$ by the above result. Now to show $E^* \subseteq E$, we show that each $E_i \subseteq E$. Clearly $E_0 \subseteq E$. Suppose $E_i \subseteq E$ and let $\gamma \in E_{i+1}$. If $\gamma \in E_i$, then clearly $\gamma \in E_{i+1}$. Otherwise there is a default $\alpha \,\&\, M\beta \vdash \gamma$ such that $\alpha \in E_i$ and $M\beta \in E$. Hence $\alpha \,\&\, M\beta \in E$, and so $\gamma \in \Gamma(E) = E$. So $E_{i+1} \subseteq E$.

For the converse, we can suppose that $E = E^*$ and we must show that E is an extension of the default theory $<D, W>$. We show that each $E_i \subseteq \Gamma(E)$, so that then $E \subseteq \Gamma(E)$, and since $\Gamma(E) \subseteq E$ by the above result, we will have $\Gamma(E) = E$ and so E will be an extension. Clearly $E_0 \subseteq \Gamma(E)$. Suppose $E_i \subseteq \Gamma(E)$, and let $\gamma \in E_{i+1}$. If $\gamma \in E_i$, then since $E_i \subseteq \Gamma(E)$, we have $\gamma \in \Gamma(E)$. Otherwise there is a default $\alpha \,\&\, M\beta \vdash \gamma$ such that $\alpha \in E_i$ and $M\beta \in E$. Now since $E_i \subseteq \Gamma(E)$, $\alpha \in \Gamma(E)$, so $\gamma \in \Gamma(E)$. So $E_{i+1} \subseteq \Gamma(E)$. □

An extension is a set of formulae in E (which is coherent). To relate extensions to SE models, we thus need to convert the E sets into SE sets. By doing this, we will gain completeness of the set of SE formulae. Then we can show that the consistent extensions correspond to SE models.

Definition. The *completion* of an extension E is the set of SE sentences:

$$\mathsf{SE}(E) = \mathrm{Th}_{\mathsf{SE}}(\{c : \phi \mid \phi \in E\} \cup \{\sim c : \phi \mid \phi \notin E\}).$$

Proposition. The SE models for the completion of E are in one-one correspondence with the consistent extensions E of a default theory.

Proof. SE(E) is coherent because if $\sim\phi \in E$, $\sim\mathsf{M}\phi \in E$ by SE deductive closure. Similarly, if $\phi \notin E$, $c : \mathsf{M}\sim\phi \in$ SE(E). The result then follows from the correspondence lemma. □

As noted above, our reconstruction enables us to have beliefs of the form $\mathsf{M}\phi$, since consequents of defaults can be arbitrary E formulae. As an example of the use of such defaults, suppose we wish to remain agnostic about ψ, given information ϕ. We would have the default $\phi \vdash \mathsf{M}\psi \,\&\, \mathsf{M}\sim\psi$. So (in the absence of any other defaults) any extension containing ϕ would contain both $\mathsf{M}\psi$ and $\mathsf{M}\sim\psi$, so could contain neither ψ nor $\sim\psi$. However, our method has the severe limitation that proofs in SE now have no natural relation to entailment over a class of SE models. That is, the only meaningful deduction is that carried out *within* a single extension of an SE default theory. This deficiency is also a weakness of Reiter's Default Logic.

3.4 Autoepistemic Logic

Finally, we look at Moore's [1985] Autoepistemic Logic, which appeared much later than the above systems, and attempted to diagnose and correct some of the problems with McDermott's [1982] nonmonotonic logics: perhaps the main problem that Moore addressed was the collapse of nonmonotonic S5 to ordinary S5. He also proposes an interpretation of nonmonotonic reasoning interesting in its own right: nonmonotonic reasoning is the reasoning of an ideally rational agent about its own beliefs. For the belief operator, Moore uses L, the dual of M: $\mathsf{L}\phi$ means 'the agent believes ϕ', or '$\sim\phi$ is not consistent with the agent's beliefs'. Thus formulae such as $\mathsf{M}\mathit{fly} \rightarrow \mathit{fly}$ that generate nonmonotonic conclusions are written as $\sim\mathsf{L}\sim\mathit{fly} \rightarrow \mathit{fly}$, and interpreted as 'it is possible to infer *fly* from the fact that $\sim$*fly* is not believed'.

The semantics of Autoepistemic Logic is given in terms of ordinary propositional calculus interpretations. Moore defines an *autoepistemic interpretation* of a theory T to consist of a PC interpretation for the language of T where formulae of the form $\mathsf{L}\phi$ are counted as separate propositions. There is one further condition: $\mathsf{L}\phi$ must be true iff ϕ is in T. Thus the autoepistemic theory itself determines the interpretation of all the formulae of the form $\mathsf{L}\phi$, and the only variation amongst interpretations is in the truth value assigned to the atomic formulae. An autoepistemic interpretation of T is an autoepistemic model of T if all the formulae of T are satisfied.

Moore defines soundness with respect to a set of premises: a theory T is said to be *sound* with respect to premises A iff every autoepistemic interpretation of T that satisfies A is an autoepistemic model of T. Thus sound theories are those such that an agent's beliefs are true whenever all its premises are true. A theory is *complete* if it includes all the formulae that are satisfied in all of its models. Two important notions are those of stability and groundedness of theories. A *stable* autoepistemic theory is one that is deductively closed with respect to PC, contains $\mathsf{L}\phi$ whenever it contains ϕ, and contains $\sim\mathsf{L}\phi$ whenever it does not contain ϕ. An autoepistemic theory T is *grounded* in premises A iff every formula of T is included in the deductive closure of the set $A \cup \{\mathsf{L}\phi \mid \phi \in T\} \cup \{\sim\mathsf{L}\phi \mid \phi \notin T\}$. Moore then shows (i) that the stable autoepistemic theories are exactly those that are complete, and (ii) that the autoepistemic theories grounded in their premises are exactly those that are sound with respect to their premises.

We now reconstruct Autoepistemic Logic in SE, the aim being to have SE models corresponding to autoepistemic models. For convenience, we add the formulae $\mathsf{L}\phi$ to the language of SE, with $\mathsf{L}\phi \equiv \sim\mathsf{M}\sim\phi$, so that the beliefs of an agent represented in a hierarchy of states are those formulae ϕ such that $\sim\phi$ is not satisfied in any of the states in the hierarchy. Thus we immediately have that $\mathsf{L}\phi$ is satisfied at a state iff ϕ is. Our reconstruction will be successful if the only consistent sets of SE sentences are those that are stable and grounded in their premises. Stability is guaranteed by making a slight modification to the SE axioms; the groundedness condition is built in as a constraint on theories.

To guarantee stability, we must modify the definition of coherence and its counterpart, the SE axiom (E4). The reason for this is, as Moore [1985] notes, that in McDermott and Doyle's [1980] and McDermott's [1982] nonmonotonic logics it is impossible to infer $\mathsf{L}\phi$ from ϕ – whereas this seems an intuitively reasonable property of the set of beliefs of an ideally rational agent. But since in SE, M is interpreted as consistency, this property does not hold in SE either. Indeed in SE, we do not want to infer $\sim\mathsf{M}\sim\phi$ from ϕ: consider the example of $\phi \equiv \mathsf{M}p$, where p is atomic. $\mathsf{M}p$ says that p is consistent; $\sim\mathsf{M}\sim\mathsf{M}p$ says that $\sim\mathsf{M}p$ is not consistent. But clearly $\sim\mathsf{M}p$ is consistent, because $\sim p$ is consistent. More precisely, with respect to an information state agnostic about p, both p and $\sim p$ are consistent, so $\mathsf{M}p$ holds, but in the accessible information state satisfying $\sim p$, $\sim\mathsf{M}p$ holds, so $\mathsf{M}\sim\mathsf{M}p$ holds in the original state – in direct contradiction of the desired inference. This example shows that the formula $\mathsf{M}\phi \rightarrow \sim\mathsf{M}\sim\mathsf{M}\phi$, i.e. $\mathsf{M}\phi \rightarrow \mathsf{LM}\phi$, i.e. the (S5) axiom, is not valid in E. But the inference of $\mathsf{L}\phi$ from ϕ must hold if we are to reconstruct Autoepistemic Logic.

To support our reconstruction, we modify the system SE as follows. The only change is to the SE axiom (E4) and its counterpart, the definition of coherence: the restriction that ϕ be non-modal has been dropped. The revised definitions are written below, with the changes in bold.

(E4) $c : \sim\phi \rightarrow c : \sim\mathsf{M}\phi$, **for ϕ an arbitrary E formula.**

Definition. A set Γ_c of E formulae is *coherent* if whenever Γ_c contains $\sim\phi$ **for ϕ an arbitrary E formula,** Γ_c contains $\sim\mathsf{M}\phi$.

Now we have the following result, where from now on we use SE and E to mean those systems with the revised definitions of (E4) and coherence.

Proposition. Γ is a consistent set of SE sentences iff Γ_c (the set of ψ such that $c:\psi \in \Gamma$) is stable.

Proof. The modifications to (E4) and the definition of coherence give us a new version of the correspondence lemma. By this lemma, if Γ is consistent, Γ_c is deductively closed, coherent and complete. Coherence says that if $\phi \in \Gamma_c$, then $\sim M\sim\phi \in \Gamma_c$, i.e. $L\phi \in \Gamma_c$. Completeness says that if $\phi \notin \Gamma_c$, $M\sim\phi \in \Gamma_c$, i.e. $\sim L\phi \in \Gamma_c$. Thus Γ_c is stable. Conversely, let Γ_c be stable. Then clearly Γ_c is coherent and complete since our definitions are equivalent to that of stability. □.

Definition. An SE theory Γ is *grounded* in A if Γ is the set of SE consequences of $c:A \cup \{c:L\Gamma_c\} \cup \{c:\sim L\overline{\Gamma}_c\}$, where $\overline{\Gamma}_c$ is the set of ψ such that $\psi \notin \Gamma_c$. The notation $c:S$ is used for the set of $c:\psi$ such that $\psi \in S$.

Definition. An SE theory Γ is *sound* with respect to premises A if every SE interpretation of Γ that is a model of $c:A$ is a model of Γ.

Proposition. Γ is sound with respect to premises A iff Γ is grounded in A.

Proof. Identical to Moore's: the change from propositional models to SE models has no effect. □

We now have a reconstruction of Autoepistemic Logic, and by looking at the formulae that hold at all information states, have a logic of belief. Recall that Moore [1985] recommends using 'weak S5' (S5 without T, otherwise known as K45) as the basis of his logic. Weak S5 consists of:

(K) $L(\phi \rightarrow \psi) \rightarrow (L\phi \rightarrow L\psi)$,
(4) $L\phi \rightarrow LL\phi$,
(5) $\sim L\phi \rightarrow L\sim L\phi$,
(N) If $\vdash \phi$, infer $L\phi$.

Bell [1989] observes that consistency of beliefs is a reasonable requirement, so suggests that the belief logic underlying Autoepistemic Logic is KD45:

(D) $L\phi \rightarrow \sim L\sim\phi$.

Now because our reconstruction of Autoepistemic Logic is based on hierarchies of information states where M is interpreted as consistency, we do not have the (S5) axiom holding in all information states. Indeed, the (S5) axiom is inconsistent with our definition of completeness, as can be seen from the example above that motivated our modifying SE in the first place. But the system E (our logic of belief) satisfies all of (K), (D), (4) and (N), and also some additional postulates concerning belief which are consequences of the axioms (E3), (E6) and (E7). (E3) is $M\sim\phi \leftrightarrow M\sim M\phi$, i.e. $M\phi \leftrightarrow ML\phi$ or $L\phi \leftrightarrow LM\phi$. The easiest way to make sense of these formulae is to read L as 'believes' and M as 'is consistent'. Thus (E3) says that (i) and (ii) are beliefs of the agent, where we have (i) ϕ is consistent (with the agent's beliefs) iff the fact that the agent believes ϕ is also consistent (with its current beliefs), and (ii) the agent believes ϕ iff it believes that ϕ is consistent (with its own beliefs). Both these postulates concerning belief and consistency seem intuitively reasonable. (E6) says that if ϕ is a modal formula, $\phi \vee \psi$ cannot be a belief unless either ϕ or ψ is a belief. As a concrete example, let ϕ be Lp and ψ be q. Then $Lp \vee q$ can only be a belief if p or q is a belief (because Lp is a belief iff p is a belief); hence if Lp is a belief or q is a belief. (E7) written with the belief operator is $L(\phi \rightarrow L\phi)$, which says that it is a belief that $\phi \rightarrow L\phi$, or, it is a belief that if ϕ is a belief of the agent, the agent believes ϕ. This asserts nothing more than the fact that it is a belief of the agent that an autoepistemic theory contains $L\phi$ if it contains ϕ. So E is strictly stronger than KD4, while it omits the (S5) axiom.

We can also see on the basis of our reconstruction of Autoepistemic Logic, that our earlier reconstruction of NMLI was misguided precisely because we had no groundedness condition. For example, with the theory $M\phi \rightarrow \phi$, we had two fixed points $\{\phi, M\phi\}$ and $\{\sim\phi, \sim M\phi\}$. But the second of these theories is not grounded in its premises, so with our reconstruction of Autoepistemic Logic, we have, using the definition proposed by Moore, overcome this difficulty with reinterpreting NMLI. Unfortunately, by introducing this constraint on theories, we can no longer interpret deduction in SE as entailment over autoepistemic models, because this constraint is not representable in SE. We also note that by simply replacing groundedness by Konolige's [1987] *strong groundedness*, we have, using Konolige's correspondence between Default Logic and Autoepistemic Logic, that the SE models of a default theory are exactly the autoepistemic models of the corresponding theory.

4 Conclusion

We have defined a logic of consistency, SE, and reconstructed some well-known nonmonotonic logics in it. The initial motivation was that these logics all employed some notion of consistency, yet in each case, a different notion was being employed. In summary, our reconstructions of the nonmonotonic logics have the following properties. In the cases of Non-Monotonic Logics I and II, we can interpret deduction in SE as entailment over classes of models, but as we have seen, the logic SE is by itself too weak to generate intuitively correct conclusions. With Default Logic and Autoepistemic Logic, we have had to sacrifice the aspect of deduction corresponding to entailment: in Default Logic because all reasoning is carried out within an extension, and in Autoepistemic Logic because of a constraint imposed on theories that is not representable in SE. The overall conclusion is that all of these methods are too weak to capture nonmonotonic reasoning as deduction in a logical system. From NMLI and NMLII, we learn that conditions stronger than pure consistency with beliefs are required. Both Reiter and Moore use non-logical constraints to enforce such conditions: Reiter using inference rules instead of logical implications, Moore using an appeal to soundness with respect to a set of premises. Furthermore, because M can no longer be interpreted strictly as consistency, we have lost the intuition about the sorts of default or autoepistemic rules that are appropriate for nonmonotonic reasoning. But our reconstructions have clarified the nature of consistency being employed in the various nonmonotonic logics, and in the case of Autoepistemic Logic, enables us to provide a new logic of belief incorporating a notion of consistency.

Acknowledgements

This work is built upon Ph.D. research carried out in the Department of Computer Science at the University of Essex in the U.K: I gratefully acknowledge the influence on my thinking of my supervisor, Prof. Raymond Turner. More recently, discussions within the Knowledge Systems Group in the Basser Department of Computer Science at the University of Sydney and especially with Dr Norman Foo have added to my understanding of nonmonotonic logics. Finally, the careful reading and insightful comments of an anonymous referee helped in clarifying both the presentation and content of the paper.

References

[Bell, 1989] Bell, J. 'Nonmonotonic Reasoning, Nonmonotonic Logics and Reasoning About Change.' Technical Report CSM-126, Dept. of Computer Science, University of Essex, 1989.

[Davis, 1980] Davis, M. 'The Mathematics of Non-Monotonic Reasoning.' *Artificial Intelligence*, **13**: 73-80, 1980.

[Etherington, 1987] Etherington, D.W. 'A Semantics for Default Logic.' *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 495-498, Milano, Italy, August 1987.

[Hughes and Cresswell, 1968] Hughes, G.E. & Cresswell, M.J. *An Introduction to Modal Logic*. Methuen, London, 1968.

[Konolige, 1987] Konolige, K. 'On the Relation Between Default Theories and Autoepistemic Logic.' *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 394-401, Milano, Italy, August 1987.

[McDermott, 1982] McDermott, D.V. 'Nonmonotonic Logic II: Nonmonotonic Modal Theories.' *Journal of the ACM*, **29**: 33-57, 1982.

[McDermott and Doyle, 1980] McDermott, D.V. & Doyle, J. 'Non-Monotonic Logic I.' *Artificial Intelligence*, **13**: 41-71, 1980.

[Moore, 1985] Moore, R.C. 'Semantical Considerations on Nonmonotonic Logic.' *Artificial Intelligence*, **25**: 75-94, 1985.

[Reiter, 1980] Reiter, R. 'A Logic for Default Reasoning.' *Artificial Intelligence*, **13**: 81-132, 1980.

[Reiter, 1987] Reiter, R. 'Nonmonotonic Reasoning.' *Annual Reviews of Computer Science*, **2**: 147-186, 1987.

[van Fraassen, 1966] van Fraassen, B.C. 'Singular Terms, Truth-value Gaps, and Free Logic.' *Journal of Philosophy*, **63**: 481-495, 1966.

[Wobcke, 1988] Wobcke, W.R. *A Logical Approach to Schema-Based Inference*. Ph.D. Thesis, Dept. of Computer Science, University of Essex, 1988.

[Wobcke, 1989] Wobcke, W.R. 'A Partial Semantics for Nonmonotonic Logics.' Technical Report 348, Basser Dept. of Computer Science, University of Sydney, 1989.

Dialectics and Specificity: Conditioning in Logic-based Hypothetical Reasoning (Preliminary Report)

David Poole
Department of Computer Science,
The University of British Columbia,
Vancouver, B.C., Canada V6T 1W5
poole@cs.ubc.ca

Abstract

In this paper we start with defaults as possible hypotheses and prediction as membership in all extensions. It is argued that this is too conservative and does not allow many intuitive answers. We show how viewing membership in all extensions as a form of dialectic, and adding a notion of conditioning can produce more intuitive answers. Defaults are possible hypotheses for a logical argument that contain a pragmatic component that is a context in which we know the default is applicable. This context is used to ignore counter arguments that follow from the context of the default. The conditioning that is presented is very close to the irrelevance that Geffner added to ϵ-semantics, and the resulting solutions turn out to be very similar.

1 Introduction

When considering default knowledge, there is a very strong notion that we should prefer more specific knowledge over more general knowledge [Touretzky, 1986, Poole, 1985, Loui, 1987, Geffner, 1988]. In probability theory this is accomplished by conditioning [Pearl, 1988]. In this paper, we show how a form of conditioning can be added to a logic-based hypothetical reasoning system. The resulting system is simple, can be easily implemented and solves many problems in a natural, straight-forward manner.

This work uses the first order predicate calculus; default reasoning is accomplished by allowing the assumption and criticism of premises in logical arguments. The use of conditioning has been inspired by probability, particularly the work of [Pearl, 1988, Geffner, 1988, Neufeld and Poole, 1988].

1.1 Logic-based Hypothetical Reasoning

Monotonicity has often been cited as a problem with using logic as a basis for commonsense reasoning. In [Poole, 1988, Poole, 1989b] it was argued that instead of deduction from our knowledge, reasoning should be viewed as a process of theory formation. In [Poole, 1988] it was shown how default reasoning can be viewed in this way by treating defaults as possible hypotheses that can be used in an explanation. In [Poole, 1989b] it was shown how membership in all extensions can form the basis for prediction and can be implemented as a process of dialectics.

1.2 Dialectics

The idea behind dialectics [Loui, 1990] is that a conclusion is reached by a process of argumentation. One agent comes up with an argument for a proposition; another agent can criticise the argument by coming up with a counter argument. In the Theorist framework all of the arguments are valid deductions; the premises are background knowledge, knowledge of the case at hand and assumptions.

One particularly appealing framework [Poole, 1989b] is where there are two agents. One agent finds arguments for a proposition. The other agent tries to either dismiss the argument out of hand (by showing it is inconsistent), or create an argument against the premises of the first agent's arguments. This idea is developed in more detail in section 2.2.

This implements membership in all extensions which is (propositionally, at least) equivalent to circumscription [Etherington, 1988]. This dialectical implementation [Poole, 1989b] provides an abstract specification of recent implementations of circumscription [Przymusinski, 1989, Ginsberg, 1989, Inoue and Helft, 1990]. In this paper it is argued that this notion of prediction is too restrictive, but is a good starting point for different argument forms.

1.3 Background and Contingent Knowledge

Consider the following example:

Example 1.1 Suppose we have as defaults "birds fly", "emus don't fly", and as facts "emus are birds" and "Tweety is an emu". There is a very strong preference for concluding "Tweety doesn't fly" based on specificity [Touretzky, 1986, Poole, 1985, Loui, 1987, Thomason and Horty, 1988]. We prefer to use the more specific knowledge about emus over the more general knowledge about birds.

The instances of the facts that are relevant to the conclusion are

$$emu(tweety) \wedge (emu(tweety) \Rightarrow bird(tweety)) \quad (1)$$

Using the same defaults, if we change the facts by swapping the role of *emu* and *bird* the answer should be, by symmetry, that Tweety does fly (i.e., the opposite of the previous conclusion). The instance of the facts used would be

$$bird(tweety) \wedge (bird(tweety) \Rightarrow emu(tweety)) \quad (2)$$

It is important to notice that formulae (1) and (2) are logically equivalent. Notice also that we have only talked about the facts and not about the defaults.

This seems to indicate that defining specificity, with logically equivalent instances of facts treated identically, is impossible. The first reaction is that these are different because the implication is an instance of a fact, and the equivalence does not hold between the facts, only between the instances of the facts. There are, however, good reasons why this is more than a syntactic distinction [Poole, 1990].

There seems to be a qualitative difference between the facts "emus are birds" and "tweety is an emu". The first is a fact that we would not like to consider being false (we would not consider the question "what if emus were not birds"), the second is one we may consider being false (e.g., we could conceive of the situation where Tweety was a sparrow).

This indicates that we should partition the facts into *background facts* and the *contingent facts* [Poole, 1985, Delgrande, 1988, Geffner, 1988]. This distinction is similar to the distinction between the network and markers in marker passing systems such as NETL [Fahlman, 1979], to the difference between the probabilistic knowledge (such as $p(A|B) = 0.345$) and the conditioning knowledge (the B in the preceding equation) in probability theory [Pearl, 1988], and to the difference between background knowledge and observations in abduction [Popl, 1973, Poole, 1989b].

1.4 Conditioning and Contexts

The final piece of the jigsaw is the notion of conditioning. If all we know about Tweety is that Tweety is an emu (given that we do not also have "emus do fly"), there is a very strong tendency to want to conclude that Tweety doesn't fly from the default "emus don't fly".

The intuition behind conditioning that will be used is that if "p's are q's" is a default and if we know $p(c)$, then all of the objections that could be raised about $q(c)$ that follow from $p(c)$ have already been taken into account when building the knowledge base. We only consider arguments against the conclusion $q(c)$ that do not already follow from $p(c)$.

This conditioning is accomplished by associating a *context* with each default, in which we know the default is applicable. Arguments against a default can be ignored if they are also arguments against the default given only the context of the default.

The notion of "context" is used, rather than, for example making a default into a pair, for a number of reasons. The first is the reluctance to invent any new connectives; part of the Theorist research is to see how far we can get inventing as little as possible. It may be the case that the appropriate context for a default is not the same as the precondition for the default (see section 5). Contexts seem to reflect a natural intuition.

The use of contexts is similar to an automatic prioritisation or cancelling of defaults, but, we will see that it has considerable advantages. One important advantage is that the sort of knowledge required to build the knowledge base is local, and so the knowledge base should be able to be built incrementally.

2 Formal Framework

2.1 Theorist Framework

Theorist is a simple framework for hypothetical reasoning.

We assume we are given a standard first order language over a countable alphabet. By a formula we mean a well formed formula in this language. By an instance of a formula we mean a substitution of terms in this language for free variables in the formula. In this paper the Prolog convention of variables starting with an upper case letter is used.

The basic definitions of Theorist are in terms of a set of closed formulae A (given as true) and a set of (possibly open) formulae H (the "possible hypotheses"). A **scenario** of (A, H) is a set D of ground instances of elements of H such that $D \cup A$ is consistent. If g is a closed formula, an **explanation** of g from (A, H) is a scenario of (A, H) which, together with A, implies g. An **extension** of (A, H) is the set of logical consequences of A together with a maximal (with respect to set inclusion) scenario of (A, H).

In [Poole, 1988] it was shown how to avoid having complex formulae as defaults by "naming" complicated defaults (similar to the use of abnormality[McCarthy, 1986]), using the name as the default and have the name implying the formula as a fact. This is done in the examples in this paper.

2.2 Membership in all extensions

It can be argued [Poole, 1989b] that predicting what is in all extensions (i.e., can be explained even if an adversary chooses the defaults) provides more satisfactory results than, for example, predicting what is in one extension. Etherington [1988] has shown that this notion of prediction corresponds (propositionally at least) to circumscription [McCarthy, 1986].

The following theorem was proved in [Poole, 1989b, theorem 2.6]:

Theorem 2.1 *g is in every extension of (A, H) if and only if there is a set $\mathcal{E}$ of (finite) explanations of g from (A, H) such that there is no scenario S of (A, H) inconsistent with every element of $\mathcal{E}$.*

Theorem 2.1 leads to the following dialectical view of membership in every extension[1][Poole, 1989b].

There are two processes $\mathcal{Y}$ and $\mathcal{N}$ that are having an argument as to whether g should be predicted. Process $\mathcal{Y}$ tries to find explanations of g. Process $\mathcal{N}$ tries to find a scenario inconsistent with all of $\mathcal{Y}$'s explanations.

In general $\mathcal{Y}$ has a set of explanations Φ (initially Φ is empty). $\mathcal{N}$ tries to find a scenario S which is inconsistent with all members of Φ (i.e., explains the conjunction of the negation of the elements of Φ). When $\mathcal{N}$ finds such a scenario S, $\mathcal{Y}$ must find an explanation of g from (S, H). Whichever process, using a complete proof procedure, gives up first loses:

- If $\mathcal{Y}$ cannot come up with an explanation based on $\mathcal{N}$'s scenario S, then g is not in all extensions (in particular g is not in any extension of S).
- If $\mathcal{N}$ cannot come up with a scenario inconsistent with all of $\mathcal{Y}$'s arguments, every extension contains at least one of $\mathcal{Y}$'s arguments, and so g is in every extension.

One further refinement of theorem 2.1 can be easily proven. This corollary says that $\mathcal{N}$ only needs to choose one default from each of $\mathcal{Y}$'s explanations.

Corollary 2.2 g is in all extensions of (A, H) if and only if there is a set $\mathcal{E}$ of explanations of g from (A, H) such that there does not exist a counter argument. Scenario S of (A, H) is a counter argument if $\forall \phi \in \mathcal{E}\ \exists d \in \phi$ such that

1. $A \wedge d \models \neg S$.

The following example shows how restricted this notion of prediction is.

Example 2.3 Suppose we have the fact that emus are birds, and the defaults "birds fly", "emu's don't fly", and "if something looks like an emu, it is an emu".

This can be represented as[2]:

$$\begin{aligned}
K = \{\ & \forall X\ emu(X) \Rightarrow bird(X), \\
& \forall X\ ll\_emu(X) \wedge mbe(X) \Rightarrow emu(X), \\
& \forall X\ bird(X) \wedge bf(X) \Rightarrow flies(X), \\
& \forall X\ emu(X) \wedge enf(X) \Rightarrow \neg flies(X) \\
H = \ & \{bf(X), \\
& enf(X), \\
& mbe(X)\}
\end{aligned}$$

[1]This algorithm corresponds to an abstract specification of algorithms for computing circumscription [Ginsberg, 1989, Przymusinski, 1989]. These algorithms find all of Y's arguments and then fail on N's counter arguments [Inoue and Helft, 1990].

[2]$ll\_emu(X)$ is intended to mean "X looks like an emu".

Using membership in all extensions as a basis for prediction, we do not predict $\neg flies(tweety)$ from

$$(K \cup \{emu(tweety)\}, H).$$

This is because of the counter argument $\{bf(tweety)\}$.

This seems like a peculiar objection to $enf(tweety)$ as it is a counter argument for any emu.

Similarly we do not predict $emu(tweety)$ from $(K \cup \{ll\_emu(tweety)\}, H)$, due to the counter argument

$$\{bf(tweety), enf(tweety)\}$$

which again, always holds whenever the default is applicable.

The objection to the conclusion of $\neg flies(tweety)$ from $(K \cup \{ll\_emu(tweety)\}, H)$, namely $\{bf(tweety)\}$, is also a peculiar objection.

The proposed "solutions" to such problems, namely using cancellation axioms [McCarthy, 1986, Poole, 1988] and providing global priorities [McCarthy, 1986], are unsatisfactory for a number of reasons (see section 6). In this paper an alternate solution is advanced.

3 Forcing Conditioning

If we have "emus don't fly" as a default, we want it to be used if all we know about an object is that it is an emu. Although there may be counter arguments (e.g., because it is a bird, it flies), we have taken these into account when building the knowledge base. The idea is to ignore counter arguments to "emu's don't fly" that follow just from the object being an emu. We still take into account other arguments as to why the emu should fly.

We assume that we are given the following sets

K a set of closed formulae; the "background knowledge". The knowledge that we know is always true.

G a set of closed formulae; the given knowledge about the situation being considered.

H a set of open formulae; the "possible hypotheses".

We associate with each possible hypothesis a context. The intention is that given just the context associated with a hypothesis we know the hypothesis is applicable (if consistent). A counter argument can be ignored if it is a counter argument when given only the context of the default.

The *context* of possible hypothesis h, written $\mathcal{C}(h)$ is a formula with free variables amongst the free variables of h.

The basic idea that we exploit is that some counter-arguments will be over-ridden by specificity. If S is an argument against d, i,e,

$$K \wedge G \wedge d \models \neg S$$

then S can be ignored due to specificity if

$$K \wedge \mathcal{C}(d) \wedge d \models \neg S$$

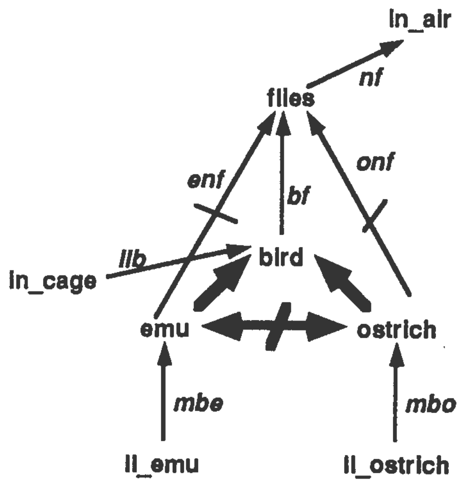


Figure 1: A diagram of the knowledge in example 3.2. Thick lines are facts, thin lines are (named) defaults.

Definition 3.1 We **predict$_1$** g if there is a set $\mathcal{E}$ of explanations of g from $(K \cup G, H)$ such that there does not exist a counter argument. Scenario S of $(K \cup G, H)$ is a counter argument if $\forall \phi \in \mathcal{E}\ \exists d \in \phi$ such that

1. $K \wedge G \wedge d \models \neg S$ and
2. $K \wedge \mathcal{C}(d) \wedge d \not\models \neg S$.

Note that prediction in this definition is a strict superset of membership in all extensions. If some formula is in all extensions, then it is predicted.

Example 3.2 Consider the following "knowledge" about birds (see figure 1):

$$
\begin{aligned}
K = \{ \ & \forall X\ emu(X) \Rightarrow bird(X) \\
& \forall X\ ostrich(X) \Rightarrow bird(X) \\
& \forall X\ \neg(emu(X) \wedge ostrich(X)) \\
& \forall X\ ll\_emu(X) \wedge mbe(X) \Rightarrow emu(X) \\
& \forall X\ ll\_ostrich(X) \wedge mbo(X) \Rightarrow ostrich(X) \\
& \forall X\ bird(X) \wedge bf(X) \Rightarrow flies(X) \\
& \forall X\ in\_cage(X) \wedge llb(X) \Rightarrow bird(X) \\
& \forall X\ emu(X) \wedge enf(X) \Rightarrow \neg flies(X) \\
& \forall X\ ostrich(X) \wedge onf(X) \Rightarrow \neg flies(X) \\
& \forall X\ flies(X) \wedge nf(X) \Rightarrow in\_air(X)\} \\
H = \{ \ & bf(X), enf(X), onf(X), mbe(X), mbo(X), \\
& llb(X), nf(X)\}
\end{aligned}
$$

The context information can be represented as

$$
\begin{aligned}
\mathcal{C}(bf(X)) &= bird(X) \\
\mathcal{C}(enf(X)) &= emu(X) \\
\mathcal{C}(onf(X)) &= ostrich(X) \\
\mathcal{C}(mbe(X)) &= ll\_emu(X) \\
\mathcal{C}(mbo(X)) &= ll\_ostrich(X) \\
\mathcal{C}(llb(X)) &= in\_bird\_cage(X) \\
\mathcal{C}(nf(X)) &= flies(X)
\end{aligned}
$$

Example 3.3 Suppose we are given

$$
\begin{aligned}
G = \{ \ & emu(tweety), \\
& in\_bird\_cage(tweety), \\
& bird(polly), \\
& in\_bird\_cage(polly)\}
\end{aligned}
$$

We predict$_1$ that tweety does not fly, as there is an explanation of $\neg flies(tweety)$, namely $\{enf(tweety)\}$. The only potential counter argument (i.e., explanation of $\neg enf(tweety)$) is $\{bf(tweety)\}$. This explanation is ignored due to specificity as

$$K \wedge enf(tweety) \wedge \mathcal{C}(enf(tweety)) \models \neg bf(tweety)$$

We do not predict$_1$ that tweety flies. There is an explanation of $flies(tweety)$, namely $\{bf(tweety)\}$, however the explanation of $\neg bf(tweety)$, $(\{enf(tweety)\})$ is not an explanation of $\neg bf(tweety)$ from the context of $bf(tweety)$, namely $bird(tweety)$.

The knowledge $in\_bird\_cage(tweety)$ provided no evidence for the flying ability of Tweety. It could be safely ignored as it was irrelevant to the conclusion.

We predict that Polly flies, as there is an argument for the flying of Polly, and no reason to doubt that argument. There is no argument for Polly not flying.

Example 3.4 Consider how example 2.3 is handled. Suppose we have the knowledge base of example 3.2, and are given

$$G = \{ll\_emu(tweety)\}.$$

There is an explanation for $emu(tweety)$, namely $mbe(tweety)$. There is one counter-argument for this, namely

$$\{bf(tweety), enf(tweety)\}$$

however, this argument follows from $\mathcal{C}(mbe(tweety))$, and so can be ignored. Thus we predict $emu(tweety)$.

We predict$_1$ $\neg flies(tweety)$. There is an explanation, namely

$$\{mbe(tweety), enf(tweety)\}$$

The same counter argument exists for $mbe(tweety)$, and can be ignored for the same reason as above. There is one explanation of $\neg enf(tweety)$, namely

$$\{mbe(tweety), bf(tweety)\}$$

This can be ignored as $bf(tweety)$ is an argument against $enf(tweety)$ given $\mathcal{C}(enf(tweety))$.

We do not predict$_1$ $flies(tweety)$. There is an explanation for $flies(tweety)$, namely

$$\{mbe(tweety), bf(tweety).\}$$

There is a counter argument (an explanation of $\neg bf(tweety)$):

$$\{mbe(tweety), enf(tweety)\}$$

which cannot be ignored as it does not follow from $\mathcal{C}(bf(tweety))$.

Example 3.5 Suppose we have the knowledge base of example 3.2 and are given that Tweety looks like an emu and also looks like an ostrich (they do look similar):

$$G = \{ll\_emu(tweety) \wedge ll\_ostrich(tweety)\}$$

There are two explanations of $bird(tweety)$, namely:

$$\{mbe(tweety)\}$$

$$\{mbo(tweety)\}.$$

There is a counter argument to each of these explanations (they are, in fact, counter arguments to each other), but there is only one potential counter argument to both explanations, namely

$$\{bf(tweety), enf(tweety), onf(tweety)\}$$

This, however is also a counter in the contexts of each default and so can be ignored.

We thus predict $bird(tweety)$. We also predict $\neg flies(tweety)$.

Example 3.6 As an interesting variation to the previous example, suppose we are given that Tweety either looks like an emu or looks like an ostrich:

$$G = \{ll\_emu(tweety) \vee ll\_ostrich(tweety)\}$$

There is one explanation of $bird(tweety)$, namely:

$$\{mbe(tweety), mbo(tweety)\}.$$

There are potential counter arguments to this explanation, namely

$$\{bf(tweety), enf(tweety)\}$$

$$\{bf(tweety), onf(tweety)\}$$

These, however can be ignored due to specificity. We thus predict $bird(tweety)$. We also predict $\neg flies(tweety)$.

These examples show the robustness of the definition of specificity.

It is interesting to consider how this definition handles the qualitative lottery paradox [Poole, 1989a] that is problematic for many systems. In [Poole, 1989a] it was shown that there is a conflict between the "one step default property" (conditioning in this paper) and conjunctive closure. It was argued that conjunctive closure was the less intuitive property.

Example 3.7 The general form of the qualitative lottery paradox given in [Poole, 1989a] can be expressed as:

$$\begin{aligned} K \;=\{\; & \forall X\ b(X) \wedge d_i(X) \Rightarrow c_i(X), \text{ for } i = 1..n, \\ & \forall X\ \neg(c_1(X) \wedge \ldots \wedge c_n(X))\} \\ H \;=\{\; & d_i(X), \text{ for } i = 1..n\} \end{aligned}$$

$$\mathcal{C}(d_i(X)) \;=\; b(X), \text{ for } i = 1..n$$

Given $b(t)$, we can predict$_1$ $d_i(t)$ (and so $c_i(t)$) for any i. We predict the conjunctions of these conclusions while they are consistent. For example, we predict$_1$

$$c_1(t) \wedge \ldots \wedge c_{j-1}(t) \wedge c_{j+1}(t) \wedge \ldots \wedge c_n(t)$$

for each j. The reason is that the only argument against each $d_i(t)$ is

$$\{d_1(t), \ldots, d_{i-1}(t), d_{i+1}(t), \ldots, d_n(t)$$

and this is an argument against d_i given only the context of the default.

We do not however predict the conjunction of all of the $c_i(t)$, as this is inconsistent and so cannot even be explained.

4 Refinement of Conditioning

Example 4.1 Consider the following facts and defaults:

$$\begin{aligned} K \;=\{\; & uni\_student(X) \wedge usa(X) \Rightarrow adult(X), \\ & uni\_student(X) \wedge usne(X) \Rightarrow \neg employed(X), \\ & adult(X) \wedge ae(X) \Rightarrow employed(X)\} \\ H \;=\{\; & usa(X), usne(X), ae(X)\} \end{aligned}$$

$$\begin{aligned} \mathcal{C}(usa(X)) &= uni\_student(X) \\ \mathcal{C}(usne(X)) &= uni\_student(X) \\ \mathcal{C}(ae(X)) &= adult(X) \end{aligned}$$

Using the previous definition of prediction, given $uni\_student(fred)$, we predict

$$adult(fred) \wedge \neg employed(fred)$$

However, given

$$uni\_student(fred) \wedge adult(fred)$$

we do not predict $\neg employed(fred)$. The counter argument, $ae(fred)$ cannot be ignored. While we cannot *prove* $\neg ae(fred)$ from any default and its context, we can *predict* $\neg ae(fred)$ from the context of either default.

Example 4.2 Suppose we are given the background knowledge of example 3.2, and the contingent knowledge,

$$G = ll\_emu(tweety) \wedge \neg in\_air(tweety)$$

There is an explanation of $emu(tweety)$, namely by assuming

$$\{mbe(tweety)\}$$

There is an explanation of $\neg emu(tweety)$, by assuming

$$\{bf(tweety), nf(tweety)\}$$

The negation of this counter argument is not *proven* from the context of any default and that default, but $\neg bf(tweety)$ is *predicted* from the context of $mbe(tweety)$.

This leads us to the next definition of prediction which allows us to predict even more. The idea is to extend the definition so that a counter argument needs to just predict the negation of the defaults. This is defined recursively to ensure that the definition is well-grounded.

Definition 4.3 We **predict**$_i$ g given G if there is a set $\mathcal{E}$ of explanations of g from $(K \wedge G, H)$ such that there does not exist a counter argument. Scenario S of $(K \wedge G, H)$ is a counter argument if $\forall \phi \in \mathcal{E}\ \exists d \in \phi$ such that

1. $K \wedge G \wedge d \models \neg S$ and
2. we do not predict$_{i-1}$ $\neg S$ given $\mathcal{C}(d) \wedge d$.

We **predict**$_0$ g given A if $K \wedge A \models g$.

Definition 4.4 We **predict** g given G if there is some i such that we predict$_i$ g given G.

In this definition predict$_1$ is the same as the previous definition; each higher integer allows us to predict more.

In example 4.1 we predict$_2$ $\neg employed(fred)$ given

$$uni\_student(fred) \wedge adult(fred)$$

In example 4.2 we predict$_2$ $emu(tweety)$ given

$$ll\_emu(tweety) \wedge \neg in\_air(tweety)$$

5 Pragmatics

Contexts are intended to be the cases under which we know the assumption is applicable. The "normal case" is where the default "p's are q's" is represented as the fact

$$\forall X\ p(X) \wedge d(X) \Rightarrow q(X)$$

with the default $d(X)$ and the context information

$$\mathcal{C}(d(X)) = p(X)$$

There is nothing in the theory to force this use of contexts. There are two extremes of contexts that are interesting. If $\mathcal{C}(d)$ is uniformly $false$, prediction becomes equivalent to membership in one extension (as all counter arguments are ignored). If $\mathcal{C}(d)$ is uniformly $true$, prediction is equivalent to membership in all extensions.

Contexts can be used in order to carry out a powerful form of preference for defaults, of which specificity is only one (albeit natural) instance.

One pragmatic idea is that if "a's are c's" and "b's are not c's", then we have a conflict if we know something is both an a and a b. If we prefer the first default over the second, we want to say that the first default is applicable even if b were true. This is done by:

$$\begin{aligned} K &= \{\ a \wedge d_1 \Rightarrow c \\ &\qquad b \wedge d_2 \Rightarrow \neg c\} \\ H &= \{\ d_1, d_2\} \\ \mathcal{C}(d_1) &= a \wedge b \\ \mathcal{C}(d_2) &= b \end{aligned}$$

If we are given a, we predict c. If we are given b, we predict $\neg c$. If we are given $a \wedge b$ we predict c, using assumption d_1; the counter argument to d_1, namely d_2 is always a counter argument to d_1 given the context of d_1.

One more thing needs to be done if the implication is of a causal type. If we are given a, we don't want to use d_1 to say that c is true and then use d_2 to say that b was not true. Such counter-intuitive reasoning can be prevented by adding the reasonable fact that the defaults should not both be used, that is, $\neg(d_1 \wedge d_2)$. This is a simplistic way to handle causal reasoning (see [Geffner, 1989] for a more sophisticated theory), but is good enough, with specificity, to handle some tricky examples:

Example 5.1 (Geffner, 1989) Suppose we get up in the morning and find that we have left the lights on in the car and want to determine whether the car will start. We are given that the car normally starts if we turn the key, and normally does not start if the battery was flat (even if we turn the key), and that the battery is flat, by default if the lights were on. Following the above methodology, this can be stated as

$$\begin{aligned} K = \{\ & turn\_key \wedge key\_starts \Rightarrow starts \\ & batt\_flat \wedge batt\_prevents \Rightarrow \neg starts \\ & \neg(batt\_prevents \wedge key\_starts) \\ & lights\_were\_on \wedge drained \Rightarrow batt\_flat\} \\ H = \{\ & key\_starts, batt\_prevents, drained\} \end{aligned}$$

$$\begin{aligned} \mathcal{C}(key\_starts) &= turn\_key \\ \mathcal{C}(batt\_prevents) &= batt\_flat \wedge turn\_key \\ \mathcal{C}(drained) &= lights\_were\_on \end{aligned}$$

Notice that the use of context allows us to say that the *batt_prevents* default is applicable even if we turn the key.

If we are given just *turn_key*, we predict *starts*, as the only counter argument can be ignored by specificity.

If we were given

$$turn\_key \wedge lights\_were\_on$$

we predict $batt\_flat$ and $\neg starts$. There are no counter arguments to *drained*, and the counter argument to *batt_prevents* is ignored by specificity.

Example 5.2 (Hanks and McDermott, 1986) Consider the celebrated "Yale Shooting Problem"[3]. We will follow the methodology given above and make one slight change to make it only a default that Fred dies if shot with the gun loaded (as she may be wearing a bullet proof vest).

$$\begin{aligned} K \;=\{ & \; loaded(1) \wedge lp(1) \Rightarrow loaded(2), \\ & \; alive(2) \wedge ap(2) \Rightarrow alive(3), \\ & \; shoot(2) \wedge loaded(2) \wedge sk(2) \Rightarrow \neg alive(3), \\ & \; \neg(ap(2) \wedge sk(2))\} \\ H \;=\{ & \; lp(1), ap(2), sk(2)\} \end{aligned}$$

$$\begin{aligned} \mathcal{C}(lp(1)) &= loaded(1) \\ \mathcal{C}(ap(2)) &= alive(2) \\ \mathcal{C}(sk(2)) &= shoot(2) \wedge loaded(2) \wedge alive(2) \\ G &= loaded(1) \wedge alive(2) \wedge shoot(2) \end{aligned}$$

The only thing "tricky" thing here is to include $alive(2)$ in the context of the default $sk(2)$, and to make it so that the two contradictory defaults are not both used.

To consider why $\neg alive(3)$ is predicted, there is one explanation for it, namely

$$\{lp(1), sk(2)\}$$

There are no counter arguments to $lp(1)$, and the only counter argument to $sk(2)$ is $\{ap(2), lp(1)\}$, which can be ignored due to specificity as $\neg ap(2)$ follows from $sk(2) \wedge \mathcal{C}(sk(2))$

6 Comparison with other systems

One of the main goals of this research is to draw a bridge between those systems that treat defaults as statements of conditionals [Geffner, 1988, Delgrande, 1988], and those that treat defaults as propositional assumptions [McCarthy, 1986, Poole, 1988]. The former have nice properties with respect to specificity, but need a form of irrelevance to allow chaining and ignoring irrelevant properties. The latter ignore irrelevant details and allow chaining, but do not handle specificity well. This paper is an attempt to consider what needs to be added to the assumption based systems to allow the natural specification of specificity. The solution to the problems of specificity is also much more natural than the solution of using global priorities, particularly as no one is prepared to say where such global priorities come from or what they mean. This sort of conditioning knowledge seems like the sort of knowledge one would have about a default.

[3]The notation is changed slightly here in order to keep the example simple, but still exhibit the anomalous behaviour.

The most interesting comparison of this work is with the addition of irrelevance to ϵ-semantics. The definition of ignoring in predict$_1$ is almost identical to the definition of irrelevance in [Geffner, 1988]. Both of theses systems fail for example 4.2, and the ignoring for the general definition of prediction in this paper is almost identical to the irrelevance of [Geffner and Pearl, 1989]. The resulting systems are, however, different. For example, because we are using normal logical connectives, we can use the contrapositive of defaults. The two systems get the same result on Geffner's examples (for example the "solution" to the Yale shooting problem in example 5.2 follows a similar idea to the solution presented in [Geffner, 1988]). It seems as though there is something important about the irrelevance that is independent of the underlying probability theory.

The use of conditioning can be motivated in a similar manner to the notion of "all I know" of Levesque [Levesque, 1990]. They are, however very different. Levesque makes no distinction between background and contingent knowledge. If someone just tells us that "Tweety is an emu" we can use that as our contingent knowledge and say that this is all we know (contingently) about Tweety. As part of what Levesque "only knows" about Tweety includes all tautologies about Tweety, instances of general information (such as "$square(tweety) \Rightarrow rectangle(tweety)$") and derived information (such as $bird(tweety)$). Levesque makes no attempt to automatically use specificity.

This work should also be contrasted to the work in inheritance systems [Touretzky, 1986, Thomason and Horty, 1988, Stein, 1989]. We are trying to add a notion of specificity to a general logic system, and want the non-defeasible statement "emus are birds" to be exactly the logical statement $\forall X\ emu(X) \Rightarrow bird(X)$. This work is most closely related to the sceptical inheritance of [Stein, 1989]; both allow for membership in all extensions with a notion of specificity. This work allows for a much more expressive language than the networks used for the inheritance theory.

This work has many similarities and differences to [Poole, 1985]. In that work the important context was the context of the more general default, whereas, in this paper the important context is the one of the more specific default. The main problem with that paper was in the underlying reasoning paradigm in which the specificity was added; this problem has recently been addressed [Simari and Loui, 1990]. In [Poole, 1985], the user was not required to specify the context of the defaults, as they are in the system described in this paper. It seems to be an advantage rather than a disadvantage to be able to specify a context in which a default is known to be applicable. As shown in the previous section, this extra pragmatic knowledge can be used to advantage in many cases.

7 Conclusion

In this paper we analysed some problems that arise from prediction based on membership in all extensions. This problem was diagnosed as being due to peculiar counter arguments. A solution was proposed that is based on a very simple idea of conditioning. This is particularly nice, as the conditioning knowledge required is local to a default, and seems to be very natural (as opposed to other solutions based on cancellation or global priorities).

Acknowledgements

Thanks to Hector Geffner and Andrew Csinger for valuable discussions on the topic of this paper. This research was supported under NSERC grant OPPOO44121.

References

[Delgrande, 1988] J. P. Delgrande, "An approach to default reasoning based on first-order conditional logic: revised report", *Artificial Intelligence*, 36(1) 63-90.

[Etherington, 1988] D. Etherington, *Reasoning with Incomplete Information*, Pitman, Morgan Kaufmann.

[Fahlman, 1979] S. E. Fahlman, *NETL: A System for Representing and Using Real-World Knowledge*, MIT Press, Cambridge, MA.

[Geffner, 1988] H. Geffner, "On the Logic of Defaults", *Proc. AAAI-88*, 449-454.

[Geffner, 1989] H. Geffner, *Default Reasoning: Causal and Conditioning Theories*, Ph.D. thesis, Computer Science, UCLA.

[Geffner and Pearl, 1989] H. Geffner and J. Pearl, "A Framework to reason with Defaults", to appear *Defeasible Reasoning and Knowledge Representation*, Kluwer Publisher.

[Ginsberg, 1989] M. Ginsberg, "A circumscriptive theorem prover", *Artificial Intelligence*, 39 209-230.

[Hanks and McDermott, 1986] S. Hanks and D. McDermott, "Default reasoning, non-monotonic logics, and the frame problem", *Proc. AAAI-86*, 328-333.

[Inoue and Helft, 1990] K. Inoue and N. Helft, *Theorem Provers for Circumscription, Proc. CSCSI-90.*

[Levesque, 1990] H. Levesque, "All I Know: A Study in Autoepistemic Logic", to appear *Artificial Intelligence.*

[Loui, 1987] R. P. Loui, "Defeat among arguments: a system of defeasible inference", *Computational Intelligence*, 3(2) 100-106.

[Loui, 1990] R. P. Loui, "Ampliative Inference, Computation and Dialectic", in J. Pollock and R. Cummins (Eds.) *AI and Philosophy*, M.I.T. Press.

[McCarthy, 1986] J. McCarthy, "Applications of Circumscription to Formalising Common Sense Knowledge", *Artificial Intelligence*, 28(1) 89-116.

[Neufeld and Poole, 1988] E. M. Neufeld and D. L. Poole, "Probabilistic Semantics and Defaults", *Proceedings of the Fourth Workshop Uncertainty in Artificial Intelligence*, University of Minnesota, 275-282.

[Pearl, 1988] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988.

[Poole, 1985] D. L. Poole, "On the Comparison of Theories: Preferring the Most Specific Explanation", *Proc. IJCAI-85*, 144-147.

[Poole, 1988] D. L. Poole, "A Logical Framework for Default Reasoning", *Artificial Intelligence*, 36(1), 27-47.

[Poole, 1989a] D. Poole, "What the lottery paradox tells us about default reasoning", *Proceedings of the First International Conference on the Principles of Knowledge Representation and Reasoning*, Toronto, 333-340.

[Poole, 1989b] D. Poole, "Explanation and Prediction: An Architecture for Default and Abductive Reasoning", *Computational Intelligence*, 5(2) 97-110.

[Poole, 1990] D. Poole, "The effect of knowledge on belief: conditioning, specificity and the lottery paradox in default reasoning", Technical Report, Computer Science, University of British Columbia.

[Popl, 1973] H. Popl, "On the mechanisation of Abductive Logic", *Proc. IJCAI-73*, 147-152.

[Przymusinski, 1989] T. C. Przymusinski, "An algorithm to compute circumscription", *Artificial Intelligence*, 38(1) 49-73.

[Simari and Loui, 1990] G. R. Simari and R. P. Loui, "Confluence of argument systems: Poole's rules revisited", to appear, *3rd Workshop on Nonmonotonic Reasoning*, Lake Tahoe, June 1990.

[Stein, 1989] L. A. Stein, "Skeptical Inheritance: Computing the Intersection of Credulous Extensions", *Proc. IJCAI-89*, 1153-1158.

[Thomason and Horty, 1988] R. H. Thomason and J. F. Horty, "Logics for Inheritance Theory", *Proc. Second International Workshop on Non-Monotonic Reasoning.*

[Touretzky, 1986] D. S. Touretzky, *The Mathematics of Inheritance Systems*, Morgan Kaufmann.

Nested Default Reasoning with Priority Levels[*]

Paul van Arragon
Dept. of Computer Science, University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1
(519) 885 1211 X3389
prvanarragon@watdragon.uwaterloo.ca

Abstract

A model of how a computer user reasons is important for effective human/computer communication. This paper develops tools for reasoning about reasoning. For accuracy and flexibility, each level of reasoning incorporates reasoning by default as well as by deduction.

Nested Theorist (NT) is a simple tool based on Theorist that allows default reasoning on arbitrarily-many levels. Prioritized Nested Theorist (PNT) uses defaults of different priority to give the system's defaults more power and to remove multiple explanations. Comparisons with related work regarding speech act theory and user modeling reveal the advantages obtained by tools with defaults on distinct levels.

1 Motivation

A model of how a computer user reasons is important for effective human/computer communication. Rather than modeling a user as a fixed set of beliefs, this paper shows how to model a user as a reasoning procedure. Modeling inferences that a user might make can provide a more accurate model of how a user's beliefs change and whether a user is able to solve particular problems.

Default reasoning is useful for both the system reasoning about the user, and for the model of the user's reasoning. Defaults can be used by the *system* to build and maintain a user model. The system often must deal with incomplete knowledge about the user. Defaults provide a way to make assumptions about the user when necessary. For example, stereotypical reasoning [Rich, 1979] and belief ascription [Kass and Finin, 1987] can be implemented via default reasoning.

The *user* deals with incomplete knowledge about the domain of discourse. Hence defaults are useful to the user. When modeling a user, the system should represent the user's defaults [Joshi *et al.*, 1984]. For example, reasoning about user defaults enables the system to avoid misleading the user. The system can reason about assumptions that a user might make based on what the system tells the user, and then block those assumptions that are false.

This paper shows how to build tools to reason by default about users who reason by default. These tools are based on Theorist, in which explanations are formed from facts and defaults. The tools incorporate distinct levels of reasoning by having the system form explanations that define how a user forms explanations. Beliefs are viewed as formulae that are in an explanation.

2 Theorist

Theorist [Poole *et al.*, 1987; Poole, 1988] is a simple framework for default reasoning. Its input is two sets of formulae, called facts and defaults. Theorist uses facts and consistent defaults as premises in a logical argument. If Theorist determines that a formula g is a logical consequence of the facts and consistent defaults, we say Theorist can *explain* g with the given facts and defaults. If new facts are added later, it may be that a goal g can no longer be explained because the defaults used are inconsistent with the new facts.

Explanations in Theorist are defined in terms of two sets of formulae input by the knowledge designer:

- $\mathcal{F}$ a set of facts: closed formulae taken as true in the domain;
- Δ a set of defaults: (possibly open) formulae taken as the "possible hypotheses" in the domain.

Definition 2.1 [Poole, *et. al.*, 1987] *An* **explanation** *from a default theory* $\langle\mathcal{F},\Delta\rangle$ *of a closed formula* g *is a set* $\mathcal{F}\cup D$ *where*

1. $\mathcal{F}\cup D \models g$,
2. $\mathcal{F}\cup D$ *is consistent, and*
3. D *is a subset of ground instances of elements of* Δ

We say that g **can be explained** from $\langle\mathcal{F},\Delta\rangle$ assuming D if $\mathcal{F}\cup D$ is an explanation of g.

Our notation is similar to Prolog syntax: variables begin with an upper case letter, and functions, predicates, and constants begin with a lower case letter. We use the universal quantifier ($\forall$), and four logical connectives: implication ($\leftarrow$), conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$).

[*]Thanks to the Universities of Waterloo, British Columbia, and Western Ontario for financial support.

Example 2.2 *Using a Default*
It is a fact that David has a flu virus, and it is a default that a flu results in nausea.

$$\begin{aligned}\mathcal{F} &= \{\ flu(david)\ \}\\ \Delta &= \{\ nausea(X) \leftarrow flu(X)\ \}\end{aligned}$$

From these statements there is an explanation of $nausea(david)$ with $D = \{nausea(david) \leftarrow flu(david)\}$.

Example 2.3 *An Inconsistent Default*
It is a fact that Eric has the flu. However, it is also a fact that Eric has been drugged, and that being drugged prevents nausea.

$$\begin{aligned}\mathcal{F} = \{\ & flu(eric)\\ & drugged(eric)\\ & \forall X\ \neg nausea(X) \leftarrow drugged(X)\ \}\\ \Delta = \{\ & nausea(X) \leftarrow flu(X)\ \}\end{aligned}$$

There is no explanation of $nausea(eric)$ because $nausea(eric)$ in inconsistent with the facts. There is an explanation of $\neg nausea(eric)$ with $D = \{\}$, since $\neg nausea(eric)$ is implied by the facts.

3 Nested Theorist

Nested Theorist (NT) is a simple tool based on Theorist that allows default reasoning on arbitrarily-many levels. Suppose a system, s, is modeling a user, u. On the metalevel, Theorist is used to build and maintain a model. This corresponds to s reasoning about u. On the object level, Theorist is used as a model of how u reasons. This corresponds to s's model of u's reasoning about the world.

3.1 Metalanguage and Object Language

To define NT, we define a language for each level of reasoning. For two levels, we define two languages: an object language (OL) to express u's facts and defaults, and a metalanguage (ML) to express s's facts and defaults [Konolige, 1982]. The OL is part of the object of study of the ML, since the ML is able to refer to u's facts and defaults that are expressed by the OL.

Both the ML and OL are first-order languages. That is, they consist of constants, functions, predicates, variables, connectives, and quantifiers, combined to form formulae, and given a Tarskian semantics.[1] To refer to statements in the OL, the ML has terms that denote these sentences. For each OL constant and variable, the ML has a corresponding constant. For each OL function, predicate, connective, and quantifier, the ML has a corresponding function. This is summarized in table 1 using examples.

To refer to u's facts, defaults, and explanations, the ML has constants $\mathcal{F}_u$ and Δ_u, and a function $E_u(D)$, where D is a set of defaults used in an explanation. The ML can express that a statement is in the facts of the user using the predicate $\in$. For example,

$$p'(a') \in \mathcal{F}_u$$

[1] See [van Arragon, 1990] for details, such as how to allow statements in which the the ML and OL share a variable, and how to deal with quantifying into a default.

OL sentences	*ML terms*
$r(f(a))$	$r'(f'(a'))$
$p \leftarrow q$	$if(p', q')$
$p \wedge q$	$and(p', q')$
$p \vee q$	$or(p', q')$
$\neg p$	$not(p')$
$\forall X\ r(X)$	$forall(x', r'(x')$

Table 1: Representing OL in ML

is a ML statement expressing that $p(a)$ is a fact of u. We purposefully use the predicate $\in$ rather than stating

$$\mathcal{F}_u = \{p'(a')\}$$

because we want to allow s to have incomplete knowledge of u. It may be that s does not know the complete contents of $\mathcal{F}_u$ or Δ_u.

If we want to represent more levels, we can define new languages. For example, for a third level, we can define an object-object language (OOL). The OL must have terms that denote symbols of the OOL, and the ML must have terms that denote the symbols of the OL that denote symbols of the OOL.

Notation For ease of notation, we do not use the $\in$ predicate, but instead treat $\mathcal{F}_u$ as a predicate. For example, we replace $p'(a') \in \mathcal{F}_u$ with $\mathcal{F}_u(p'(a'))$. Similarly, we treat Δ_u and E_u^D as ML predicates in our notation. For example, $\Delta_u(p')$ means that p is in the defaults of u, and $E_u^{\{p'\}}(q')$ means that q is explained by u, assuming p.

Furthermore, instead of using the ML with all of its complexity, we replace ML terms with the corresponding OL sentence. For example, rather than stating

$$\mathcal{F}_u(forall(x', and(p'(x'), q')))$$

to say that $\forall X\ p(X) \wedge q$ is in the facts of u, we use the following simpler notation:

$$\mathcal{F}_u(\forall X\ p(X) \wedge q)$$

We use the symbol $\mathcal{F}_s$ and Δ_s to refer to the set of s's defaults.[2]

3.2 Assuming Defaults are Consistent

If s builds a model of u's reasoning where u makes assumptions to explain some formula g, there are two conditions this model should satisfy (see definition 2.1):

Condition 1: the assumptions (together with the facts) of u imply g, and

Condition 2: the assumptions of u must be consistent with the facts of u.

To show that condition 1 holds, s must show that a subset of $\mathcal{F}_u$ together with the defaults u assumes imply g. It suffices to show that a subset of $\mathcal{F}_u$ implies g because such deductive conclusions are *monotonic*. That

[2] $\mathcal{F}_s$ and Δ_s are not part of the ML described above. They are used for convenience in the examples, just as $\mathcal{F}$ and Δ are used in examples 2.2 and 2.3.

is, given that $\mathcal{F}'_u$ is a subset of $\mathcal{F}_u$, and that D is the set of assumptions of u, if $\mathcal{F}'_u \cup D \models g$ then $\mathcal{F}_u \cup D \models g$.

To show that condition 2 holds is more difficult. It requires knowledge of the *complete* set $\mathcal{F}_u$. If s discovers a new element of $\mathcal{F}_u$, then u's assumptions may no longer be consistent since default conclusions are *nonmonotonic.* That is, given that $\mathcal{F}'_u$ is a subset of $\mathcal{F}_u$, and that D is the set of assumptions of u, if $\mathcal{F}'_u \cup D$ is consistent, it does not necessarily follow that $\mathcal{F}_u \cup D$ is consistent.

Example 3.4 *Unsatisfied Condition 2*
s has two facts, that u has a fact f, and that u has a default $n \leftarrow f$:[3]

$$\mathcal{F}_s = \{ \quad \mathcal{F}_u(f) \qquad (1)$$
$$\Delta_u(n \leftarrow f) \quad \} \qquad (2)$$

s cannot explain that u explains n, because s cannot show that $n \leftarrow f$ is consistent with u's facts. (1) does not say that f is the *only* fact of u. There may be other unknown facts of u that are inconsistent with $n \leftarrow f$. For example, u may have the fact $\neg n$.

The inability to satisfy condition 2 stems from incomplete knowledge regarding the facts of u. One way to solve this problem is to always specify that s knows all of u's facts. However, the usefulness of NT would be restricted since we rarely (if ever) can obtain such complete knowledge. NT should allow s to draw conclusions despite having incomplete knowledge.

Defaults are a useful tool for reasoning with incomplete knowledge. For s to reason that u is able to use a consistent default, s can assume that that $\mathcal{F}_u \cup D$ is consistent, for some set of u's defaults D. This assumption of s is blocked if s can show that $\mathcal{F}_u \cup D$ is inconsistent. With this assumption, it is possible that u has other facts that s does not know about, so s is not assuming complete knowledge. This assumption is just sufficient to enable s to conclude that u can consistently assume D.

Reconsider example 3.4. Assuming that $\mathcal{F}_u$ is consistent with default (2) enables s to satisfy condition 2. Hence, s can show that u explains n, as desired, and it is still possible that u has facts other than f, as long as they do not conflict with (2). For example, the assumption of consistency precludes that u has a fact $\neg n$, but u may have other facts, unrelated to n and f.

3.3 Stating Object-level Facts Consistent

Theorist is intended to be used by specifying a set of facts and defaults. The facts must be consistent, or else no explanations exist. The following example shows that, for nested reasoning, it is useful to specify that the object-level facts are consistent.

Example 3.5 *Facts Explicitly Consistent*
s has two facts:

$$\mathcal{F}_s = \{ \quad \mathcal{F}_u(p) \vee \mathcal{F}_u(q) \qquad (3)$$
$$\mathcal{F}_u(\neg p) \quad \} \qquad (4)$$

[3]This is a propositional nested version of example 2.2.

From (3) and (4), s cannot explain that u can explain q. However, if s also knows that u's facts are consistent, s can explain that p is *not* in $\mathcal{F}_u$ (since $\neg p$ *is* in $\mathcal{F}_u$), and therefore s can explain that q is in $\mathcal{F}_u$, and hence that u can explain q.

3.4 Definition of NT

Definition 3.6 *An* NT *explanation from* $\langle \mathcal{F}, \Delta \rangle$ *is a Theorist explanation with the following four constraints:*[4]

1. *The language of* $\mathcal{F}$ *and* Δ *is a ML that refers to an OL (as described in section 3.1);*
2. *The metalevel, s, has a fact that each agent also forms explanations according to the definition of Theorist:*

$$\forall A \forall D \forall \alpha \;\; E_A^D(\alpha) \;\leftarrow\; \mathcal{F}_A \cup D \models \alpha \qquad (5)$$
$$\wedge \;\; \mathcal{F}_A \cup D \not\models \mathit{false}$$
$$\wedge \;\; D \subseteq \Delta_A$$

3. *s has a default that a subset of each agent's defaults are consistent. That is, for each agent A, and each set D of OL formula, s has a default:*

$$(\mathcal{F}_A \cup D \not\models \mathit{false}) \;\leftarrow\; (D \subseteq \Delta_A) \qquad (6)$$

4. *s has a fact that each agent's facts are consistent. That is, s has a fact:*

$$\forall A \;\; \mathcal{F}_A \not\models \mathit{false} \qquad (7)$$

3.5 Capabilities of NT

Example 3.7 *Defaults on Two Levels*
s has two facts:

$$\mathcal{F}_s = \{ \quad p \qquad (8)$$
$$\Delta_u(r \leftarrow q) \quad \} \qquad (9)$$

and a default:

$$\Delta_s = \{ \quad \mathcal{F}_u(q) \leftarrow p \quad \} \qquad (10)$$

s can consistently assume (10), so together with (8), s explains $\mathcal{F}_u(q)$. Using (9), s can also explain $E_u^{(\{r \leftarrow q\})}(r)$, by assuming that u can consistently assume $r \leftarrow q$.

These assumptions of s and u can be made inconsistent independently. Adding that s has a fact

$$\mathcal{F}_u(\neg q) \qquad (11)$$

makes s's assumption (10) inconsistent, since (7), (8), and (11) together conflict with (10). Adding that s has a fact

$$\mathcal{F}_u(\neg r) \qquad (12)$$

makes u's assumption (9) inconsistent, since (8), (10), and (12) together conflict with (9).

The following example shows that it is also possible to state negative knowledge in NT. Negative statements are useful for reasoning about the ignorance of a user.

[4][van Arragon, 1990] shows how this definition can be nested to allow reasoning about how the object-level agent reasons about other reasoning agents.

Example 3.8 *Negative Knowledge*

We can state that s has a fact that u cannot explain p:

$$\forall D \;\neg E_u^D(p) \qquad (13)$$

If s also has a fact that u has a fact p or a fact q,

$$\mathcal{F}_u(p) \vee \mathcal{F}_u(q) \qquad (14)$$

it follows that s can explain $E_u^{\{\}}(q)$.

4 Prioritized Nested Theorist

4.1 Conflict Between Levels

A technical problem with NT as defined above is that defaults on the metalevel are weak. If s has a fact that u has a default $p(X)$, then s can explain that u can explain $p(a)$ for an arbitrary object a. Now if s assumes that u has a fact that $\neg p(a)$ is an exception to the default $p(X)$, then s can explain that u can explain $\neg p(a)$. However, it also still follows that s can explain that u can explain $p(a)$.

Example 4.9 *Multiple Explanations*

s has a fact that u has a default $p(X)$:

$$\mathcal{F}_s = \{ \quad \Delta_u(p(X)) \quad \} \qquad (15)$$

and s has a default that u has a fact $\neg p(a)$:

$$\Delta_s = \{ \quad \mathcal{F}_u(\neg p(a)) \quad \} \qquad (16)$$

Two mutually inconsistent explanations exist. In one, s assumes (16); hence s can explain $E_u^{\{\}}(\neg p(a))$. In the other, s assumes u's default (15) is consistent; hence s can explain $E_u^{\{p(a)\}}(p(a))$.

In example 4.9, statement (15) itself does not contradict statement (16). The internal default (6), which s uses to assume that u's defaults are consistent, is in direct conflict with (15).

Let $C_A(D)$ be an abbreviation for (6). That is, $C_u(D)$ means that D, a subset of Δ_u can be consistently assumed by u. In example 4.9, for s to explain that u explains $p(a)$, s assumes

$$C_u(\{p(a)\}) \qquad (17)$$

(17) and (16) are in direct conflict. Because of this conflict, s can never block an assumption of u by assuming that u has specific knowledge of an exception.

4.2 Prioritized Defaults

Brewka [1989] has expanded Theorist to include defaults of different priority levels so that a default of higher priority can block a default of lower priority. This idea is more powerful than methods of removing multiple explanations [Poole, 1988] that do not allow one default to block another.

Example 4.10 *Priority Levels*

Let D^i be the set of defaults of priority i, where a smaller i indicates higher priority. (1 is the highest priority level.)

$$\Delta^1 = \{ \neg p(a) \} \qquad (18)$$
$$\Delta^2 = \{ p(X) \} \qquad (19)$$

Given (18) and (19), defaults of different priority, $\neg p(a)$ can be explained, but $p(a)$ cannot.

We can use this idea to solve our multiple-explanations problem. By defining (16) to have priority over (17), it is possible for (16) to block (17) so that only the desired explanation exists. We outline the definition of Prioritized Nested Theorist here.

The Need for Many Priority Levels In example 4.9, s needs two levels of priority: (16) is of high priority, and (17) is of low priority. The same holds for other levels of reasoning. For example, if u is reasoning about some agent $a1$, u needs two levels of priority.

This means s must reason about u who now has two levels of priority. This causes a new problem: s must assume that u's defaults satisfy the priority constraints. In Prioritized Theorist, a low-priority default cannot be used if a high-priority default contradicts it. But recall that s has incomplete knowledge about u. In particular, s does not know the complete set of u's high-priority defaults. Therefore, an assumption that u's low-priority defaults satisfy priority constraints is necessary.

We show in [van Arragon, 1990] that this assumption of s that priority constraints are satisfied for u may conflict with the assumption of s that u's defaults are consistent. For this reason, s needs at least three levels of priority to reason about u if u has two levels of priority.

Similarly, if we add a fourth level of reasoning (say agent $a1$ is reasoning about some agent $a2$), s needs at least four levels of priority to reason about u's three levels of priority. For this reason, we define Prioritized Nested Theorist (PNT) so that s has at least as many priority levels of defaults as there are nested levels of agents.

Metalanguage and Object Language To define PNT, we must augment the ML so that it can refer to the priority levels of defaults of u. We add to the ML constants Δ_u^i for each i to refer to the u's defaults of priority i. As before, we use Δ_u^i as a predicate. For example, $\Delta_u^3(p(X))$ means that $p(X)$ is in the set of u's defaults of priority 3.

Definition of PNT A PNT explanation from $\langle \mathcal{F}, \Delta^1, \ldots, \Delta^n \rangle$ is a Prioritized Theorist explanation with constraints similar to those in the definition 3.6 of NT. The most interesting difference is that s must not only assume that u's facts are consistent, but s must also assume that u's defaults satisfy priority constraints. The priority of these defaults depends on the number of nested levels of reasoning.

Since priority levels allow a single level to be more expressive, PNT also allows more priority levels on each reasoning level than are needed to deal with the multiple explanation problem discussed above.

4.3 Capabilities of PNT

We can now modify example 4.9 to avoid multiple explanations.

Example 4.11 *Metalevel Prioritized Defaults*

s has a fact that u has a default $p(X)$:

$$\mathcal{F}_s = \{ \quad \Delta_u(p(X)) \quad \} \qquad (20)$$

and s has a high-priority default that u has a fact $\neg p(a)$:

$$\Delta_s^1 = \{ \quad \mathcal{F}_u(\neg p(a)) \quad \} \tag{21}$$

By the definition of PNT, s has a low priority default to assume that a subset of u's defaults are consistent:

$$\Delta_s^2 = \{ \quad C_u(D) \quad \} \tag{22}$$

An instance of (22) is $C_u(\{p(a)\})$ (an assumption that $D = \{p(a)\}$ is consistent). However, this instance conflicts with (21). Therefore, s can explain $E_u^{\{\}}(\neg p(a))$, but s cannot explain $E_u^D(p(a))$ for any D.

5 Applications

NT provides a general tool for user modeling. Metalevel defaults enable building and maintaining a user model by assuming things about a user. Object level facts and defaults enable modeling a user's reasoning in a flexible way. In particular, object-level defaults are useful for reasoning about how a user ascribes beliefs to another agent [Wilks and Ballim, 1987], how a user's beliefs change when an utterance occurs (as in speech-act theory[Perrault, 1987]), and how utterances may mislead a user [Joshi *et al.*, 1984]. We consider these examples briefly.

Example 5.12 *Belief Ascription*
s can ascribe facts to u by assuming that if X is a typical fact, then X is a fact of u. For example, if s can explain that a typical fact is that the world is round (23), s can assume (24) that u has a fact that the world is round. Let the following be a fact of s:

$$typical(round(world)) \tag{23}$$

and the following be a default of s:

$$\mathcal{F}_u(X) \leftarrow typical(X) \tag{24}$$

Belief ascription can be nested by stating that u also has a default like (24). Let the following be a fact of s:

$$\Delta_u(\mathcal{F}_a(X) \leftarrow typical(X)) \tag{25}$$

Now, if s can explain that, for u, a typical fact is that the world is round, s can assume that u has a fact that the world is round, and s can explain that u assumes that agent a has a fact that the world is round.

The belief ascription assumption can be contradicted at any level by specific knowledge. For example, the following fact of s prevents u from assuming that a has a fact that the world is round:

$$\mathcal{F}_u(\mathcal{F}_a(\neg round(world)) \tag{26}$$

Note that, just as in [Wilks and Ballim, 1987], we do not have to predefine the facts and defaults on all levels, but can assume facts and defaults as necessary.

Example 5.13 *Speech Acts*
A theory of speech acts specifies how utterances affect the beliefs of the hearer. An approach that can be implemented with PNT is to have the hearer u to assume that what the speaker s says is true:

$$\Delta_u^2(X \leftarrow declares_s(X)) \tag{27}$$

If we add to (27) that u has a fact that s declares p:

$$\mathcal{F}_u(declares_s(p)) \tag{28}$$

then u assumes that p is true. However, this assumption may be contradicted if u has a higher priority default that contradicts p:

$$\Delta_u^1(\neg p) \tag{29}$$

PNT provides a flexible language for stating the relative strengths of various defaults of u based on whether u believes s to be lying, or to be an authority, and so on. For example, u may assume at high priority that if agent A declares X, and X is within A's area of expertise, then X is true:

$$\Delta_u^1(X \leftarrow declares_A(X) \wedge expertise(A, X)) \tag{30}$$

The theory of belief revision of u is built into PNT. It is based directly on Prioritized Theorist, because PNT defines explanations of u to be Prioritized Theorist explanations.

Example 5.14 *Preventing False Assumptions*
This theory of u's belief revision is useful for predicting what u will believe based on s's utterances. s can use this theory to avoid misleading u. For example, s may model that u assumes that people with the flu have nausea:

$$\Delta_u^1(nausea(X) \leftarrow flu(X)) \tag{31}$$

If s tells u that Eric has the flu, s can predict that u will assume Eric has nausea. If s knows that Eric is drugged to prevent nausea, s can warn u that Eric is an exception. Note that a theory of u's defaults is crucial to deciding what u should be told.

6 Related Work

Although other research studies reasoning about agents that use defaults [Joshi *et al.*, 1984; Perrault, 1987; Appelt and Konolige, 1988], the level distinction is not fully made elsewhere. An advantage of making a clear level distinction is that NT has more expressive power. In NT, we can state explicitly that an agent makes a default assumption.

Each of these other systems lacks the ability to express user defaults explicitly. For example, rather than expressing that $q \leftarrow p$ is a default of the user, each of these system's expresses a sort of approximation of this statement. Each can state something like, "it is a default of the system that, if the user believes p, then the user believes q." Default reasoning regarding the revision of a user's beliefs takes place solely on the metalevel, since the object level models only deduction, but lacks defaults. Not only is this a conceptual disadvantage, but also it does not work for cases where the distinction between defaults on different levels is important. By contrast, in NT, reasoning about belief revision is a natural result of viewing a user as a default reasoner. We do not have to specify metalevel axioms to express how a user's beliefs are revised.

Example 6.15 *Metalevel Versus Object Level*
Given the following in PNT,

$$\mathcal{F}_s = \{ \quad \Delta_u^1(p) \quad \} \tag{32}$$
$$\Delta_s^1 = \{ \quad \mathcal{F}_u(\neg p) \quad \} \tag{33}$$

s explains that u can explain $\neg p$ but cannot explain p.

It is difficult to translate example 6.15 so that defaults are on a single level. For example, it is not equivalent to state that s has the following two defaults:

$$\Delta_s^1 = \{ \quad \mathcal{F}_u(p) \quad \} \tag{34}$$
$$\Delta_s^1 = \{ \quad \mathcal{F}_u(\neg p) \quad \} \tag{35}$$

(34) and (35) conflict, so neither is preferred over the other. In contrast, (33) is preferred over assuming that u's default in (32) is consistent.

A more accurate translation retains the relative priority:

$$\Delta_s^2 = \{ \quad \mathcal{F}_u(p) \quad \} \tag{36}$$
$$\Delta_s^1 = \{ \quad \mathcal{F}_u(\neg p) \quad \} \tag{37}$$

In general, the translation made is to replace an object-level default of priority X with a metalevel default of priority $X + 1$. This is because, for s to reason that u uses a default of priority X, s must assume with priority $X + 1$ that u's default is consistent (according to the definition of PNT).

Interestingly, work in Speech Act theory by Appelt and Konolige [1988] uses HAEL (Hierarchic-autoepistemic Logic), which has a kind of prioritized default. However, HAEL defaults are on the metalevel only. There is no facility for specifying user defaults. Hence, HAEL is less expressive than PNT.

For example, the following two-level default

$$\Delta_s^Y(\ p \wedge \Delta_u^X(p) \ ) \tag{38}$$

for some X and Y, cannot be translated to a metalevel default of the form

$$\Delta_s^Z(\ p \wedge \mathcal{F}_u(p) \ ) \tag{39}$$

The difficulty is to decide what value Z should be. If $Z > Y$, then (39) is contradicted by a default of the form

$$\Delta_s^Y(\ \neg p \ ) \tag{40}$$

although (38) would not be contradicted. But if $Z < X - 1$, then (39) is not contradicted by a default of the form

$$\Delta_s^{X-1}(\ \mathcal{F}_u(\neg p) \ ) \tag{41}$$

although assuming that the object level default p in (38) is consistent would be contradicted. Intuitively, Z cannot store the information of two priority levels using only a single number.

Apart from these technical considerations, it is useful for conceptual reasons to distinguish between assumptions on different levels. The above translations do not permit explicit specification of object-level defaults.

7 Conclusions

The problem of user modeling can be solved by addressing four subproblems:

1. formally defining the system's reasoning about the user's reasoning,
2. implementing this formal definition,
3. outlining a strategy for applying the definition to particular problems,
4. compiling a knowledge base according to this strategy.

This paper focusses on subproblem 1. NT and PNT are tools that deal with incomplete knowledge on multiple levels of reasoning. [van Arragon, 1990] presents an implementation of NT and PNT (subproblem 2).[5] In section 5, we showed briefly three strategies for applying NT (subproblem 3). Subproblem 4 encompasses the issue of knowledge acquisition, which is beyond the scope of this paper.

The main issue that arose with subproblem 1 were how a system can conclude that a user is able to use a default despite the system's incomplete knowledge about the user. To deal with this incomplete knowledge, the system can assume that a set of the user's defaults are consistent. This gives rise to a multiple extension problem that can be solved using prioritized defaults.

Since NT is a flexible tool, it provides a framework in which a wide range of user modeling strategies can be accommodated. User modeling strategies that were developed using varying underlying formalisms can be joined in one system so that they can function together. If a strategy does not fit into the NT framework, it may be possible to further augment NT. By doing so, the defining of formal tools and the applied study of user modeling strategies can benefit each other.

One way that we have augmented NT is to enable NT to reason about agents who have limited reasoning resources [van Arragon, 1990]. Many formal definitions of limited reasoning have been proposed [Fagin and Halpern, 1988; Hadley, 1988; Konolige, 1985; Levesque, 1984]. However, none of these are easy to apply to user modeling, because the limitations are inflexible compared to the demands of user modeling where the model of each individual user may vary greatly. Limited Nested Theorist (LNT) is based on the idea that reasoning limitations can be defined by having s assume that u is able to make each inference. Conflicting evidence may block such an assumption if s knows that u's reasoning resources prevent making the inference. With LNT, the knowledge designer is free to tailor the type of reasoning resources in a flexible way.

[5]Our approach is to use metaprogramming. Each agent's reasoning process corresponds to a Theorist interpreter. To model s reasoning about u, we have a Theorist interpreter reasoning about a Theorist interpreter. A problem with this general approach is the computational overhead that arises because each step of inference on the object level takes several steps on the metalevel. Fortunately, this overhead can be removed using program transformation techniques.

Acknowledgements

Thanks to members of the LPAIG group (mostly at the University of Waterloo) especially to my supervisor David Poole for many ideas and technical help, to Randy Goebel for ideas and suggestions, to Robin Cohen, Peter van Beek, Bruce Spencer, and Scott Goodwin for comments on this and other related papers.

References

[Appelt and Konolige, 1988] Douglas E. Appelt and Kurt Konolige. A practical nonmonotonic theory for reasoning about speech acts. In *Proceedings of the Twenty-sixth annual meeting of the Association for Computational Linguistics (ACL-88)*, pages 170–178, Buffalo, NY, 1988.

[Brewka, 1989] Gerhard Brewka. Preferred subtheories: An extended logical framework for default reasoning. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 1043–1048, Detroit, MI, 1989.

[Fagin and Halpern, 1988] Ronald Fagin and Joseph Y. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34(1):39–76, 1988.

[Hadley, 1988] Robert F. Hadley. Logical omniscience, semantics and models of belief. *Computational Intelligence*, 4(1):17–30, February 1988.

[Joshi *et al.*, 1984] Aravind K. Joshi, Bonnie Webber, and Ralph M. Weischedel. Preventing false inferences. In *Proceedings of the Tenth International Conference on Computational Linguistics (COLING-84)*, pages 134–138, Stanford, CA, 1984.

[Kass and Finin, 1987] Robert Kass and Tim Finin. Rules for the implicit acquisition of knowledge about the user. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 295–300, Seattle, WA, 1987.

[Konolige, 1982] Kurt Konolige. A first-order formalisation of knowledge and action for a multi-agent planning system. *Machine Intelligence*, 10:41–72, 1982.

[Konolige, 1985] Kurt Konolige. Belief and incompleteness. In Jerry R. Hobbs and Robert C. Moore, editors, *Formal Theories of the Commonsense World*, pages 359–404. Ablex Publishing Corporation, Norwood, NJ, 1985.

[Levesque, 1984] Hector J. Levesque. A logic of implicit and explicit belief. In *Proceedings of the Fourth National Conference on Artificial Intelligence (AAAI-84)*, pages 198–202, Austin, TX, 1984.

[Perrault, 1987] C. Raymond Perrault. An application of default logic to speech act theory. Technical Report CSLI-87-90, Center for the Study of Language and Information, 1987.

[Poole *et al.*, 1987] David Poole, Randy Goebel, and Romas Aleliunas. Theorist: a logical reasoning system for defaults and diagnosis. In Nick Cercone and Gordon McCalla, editors, *The Knowledge Frontier: Essays in the Representation of Knowledge*, pages 331–352. Springer, New York, 1987.

[Poole, 1988] David Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36(1):27–47, 1988.

[Rich, 1979] Elaine Rich. User modelling via stereotypes. *Cognitive Science*, 3:329–354, 1979.

[van Arragon, 1990] Paul van Arragon. *User Modeling by Nested Default Reasoning.* PhD thesis, University of Waterloo, Waterloo, Ontario, Canada, 1990.

[Wilks and Ballim, 1987] Yorick Wilks and Afzal Ballim. Multiple agents and the heuristic ascription of belief. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 118–124, Milan, Italy, 1987.

On Heterogeneous Model-Preference Default Theories

Fabrizio Sebastiani*
Istituto di Elaborazione dell'Informazione
Consiglio Nazionale delle Ricerche
Via S. Maria, 46 - 56126 Pisa (Italy)

Abstract

Default systems based on the notion of "model-preference" have recently been proposed by Selman and Kautz to give a semantic account of the phenomena involved in default reasoning and to provide a formal justification for the limited cognitive load that default reasoning seems to require of human beings. In this paper we argue that the way these formal systems have been defined makes them inadequate for the task of reasoning in the presence of both certain information and defeasible information. We propose a modification to the original framework and argue that it formalizes correctly the interaction between these two fundamentally different kinds of information. We then show that the proposed modification has also a positive effect on the complexity of model-preference default reasoning.

1 Introduction

Default reasoning plays an important role in everyday practical reasoning. Agents, be they natural or artificial, typically face situations in which they have to act and take decisions on the basis of a body of knowledge that is far from being an exhaustive description of the domain of discourse; this is a direct consequence both of the limited capacity of their physical repositories of knowledge, and of the fact that the processes involved in the acquisition of knowledge (both from external sources—e.g. books—and internal ones—e.g. speculative reasoning) are computationally demanding and time consuming.

Nevertheless, action and decision-making often require more knowledge than our agents actually possess, thus forcing them to make up with the limited coverage of their knowledge bases by means of "default" assumptions. As the name implies, "assumptions" have an epistemic status that is far from being solid, as they can actually be invalidated by further reasoning or by the future acquisition of empirical data. These phenomena are well-known in cognitive science, and have sometimes been taken to imply that a great deal of human reasoning does not conform to the canons of "logic" and hence escapes attempts at formalization [Johnson-Laird, 1983]. Doubtless, the overall effectiveness of human reasoning testifies to the effectiveness of this modality of reasoning too: humans are much quicker at finding surrogates of missing knowledge than at actually acquiring this knowledge, either through reasoning or empirical investigation. And, above all, once these surrogates have been found, humans are much quicker at reasoning on the resulting complete, albeit epistemically shakier, description of the domain of discourse than they would be had they to rely on the smaller part of this description that they trust as being accurate *tout court*. This observation is at the heart of the recent interest that the knowledge representation community has shown in *vivid knowledge bases* [Levesque, 1986, 1988; Etherington *et al.*, 1989], i.e. exhaustive descriptions of the domain of discourse consisting of collections of atomic formulae[1]. Reasoning on these KBs, which may be considered as "analogues" of the domain being represented, is easily shown to be efficient.

It is precisely in the face of such empirical considerations that the bad computational properties of current formalisms that attempt to formalize default reasoning (such as the ones based on Circumscription [McCarthy, 1980; 1986] or on Autoepistemic Logic [Moore, 1985; Konolige, 1987]) are particularly disturbing: arguably, a formalism for default reasoning not only should account for the conclusions that agents draw in the presence of incomplete information, but it also should possess radically better computational properties than formalisms aimed at reasoning tasks at which humans are notoriously inefficient (such as e.g. first order logic in the case of deductive reasoning). Moreover, it is certainly not plausible that, in order to arrive at knowledge bases upon

*Current address: Department of Computer Science, University of Toronto, MS5 1A4 Toronto, Ontario, Canada. E-mail: fabrizio@ai.toronto.edu

[1]In formally introducing vivid KBs Levesque [1988] actually situates his discussion in the framework of the first order predicate calculus; hence, for him a vivid KB is "a collection of ground, function-free atomic sentences, inequalities between all different constants (...), universally quantified sentences expressing closed world assumptions (...) over the domain and over each predicate, and the axioms of equality". As our discussion will be situated in the framework of the propositional calculus, we will take this definition of vivid KB instead.

which fast reasoning can be carried out, humans use a dramatically inefficient reasoning method.

These considerations lead us to look with special interest at formalizations of default reasoning that emphasize computational tractability. In their recent paper "The complexity of model-preference default theories" (hereafter [MPD]), Selman and Kautz [1988] describe $\mathcal{DH}_a^+$, a tractable system for performing inferences on theories of Acyclic Horn Defaults. Their framework has the added appeal of possessing a strong model-theoretic flavour[2], which allows thorough investigations in the mechanisms underlying the inference processes. It is by means of such investigations, however, that one can discover that the systems of [MPD] have an odd behaviour when confronted with knowledge bases that consist of both certain information and default information. This is especially disturbing, as we surely would like to account for the fact that agents, although making heavy use of default reasoning, normally do also possess information upon which they rely with special confidence. Accordingly, a reasoning system should enforce a correct interaction between certain knowledge and default knowledge, and account for the different impact that the two kinds of knowledge have on the overall reasoning process. It is these considerations which inform the attempt, described in this paper, to tune model-preference default systems in such a way as to make them behave correctly with respect to the distinction between certain knowledge and defeasible knowledge.

In order to make this paper self-contained, in Section 2 we give a brief overview of $\mathcal{D}^+$, the most general system described in [MPD], of which $\mathcal{DH}_a^+$ is a tractable subset[3]. In Section 3 we argue that the two methods proposed in [MPD] for dealing with the co-presence of certain information and defeasible information in $\mathcal{D}^+$ are, for different reasons, both inadequate, and characterize two interesting classes of reasoning tasks that are mishandled by both methods; we proceed to spell out our modifications to $\mathcal{D}^+$ and to argue that the system so obtained handles well the interaction between certain knowledge and defeasible knowledge (including the two classes of reasoning tasks that had revealed problematic for the original version of $\mathcal{D}^+$). In Section 4 we examine the effects of our modifications on the computational complexity of model-preference default reasoning; such modifications allow us to establish new results for reasoning in the presence of both certain knowledge and defeasible knowledge, and to discover that the presence of certain knowledge has a beneficial effect on the efficiency of the reasoning process. Section 5 concludes.

[2]A semantics for Selman and Kautz's model-preference default systems that fully embraces the model-theoretic credo is described in [Sebastiani, 1990a].

[3]In this paper we will implicitly rule out from consideration the system $\mathcal{D}$, as its lack of commitment to any specificity ordering between defaults (see below) makes it less interesting than the other systems of [MPD]. The other systems discussed in [MPD], $\mathcal{DH}^+$ and $\mathcal{DH}_a^+$, are restrictions of $\mathcal{D}^+$ to the Horn case and to the Horn Acyclic case, respectively; all modifications that are described in this paper apply straightforwardly to these more restricted systems.

2 An overview of Selman and Kautz's system $\mathcal{D}^+$

Roughly speaking, the idea around which the systems of [MPD] revolve is that the import of a default $d \equiv \alpha \rightarrow q$ is to make a model (that is, a complete specification of what the domain of discourse is like) where both α and q are true be *preferred* to another model where α is true but q is not. By combining the effects of the preferences due to the single defaults, a set of defaults identifies a set of "maximally preferred" models; these models, isomorphic as they are to vivid knowledge bases, are meant to represent possible ways in which the agent may "flesh out" his body of certain knowledge by the addition of defeasible knowledge. For instance, according to a set of defaults such as $\{a \rightarrow b,\ b \rightarrow c\}$, the model where a, b and c are all true would be a maximally preferred model. However, the systems in [MPD] also account for the fact that a more specific default should override a less specific one, and they do so by "inhibiting", where a contradiction would occur, the preference induced by the less specific default; this is meant to prevent a set of defaults such as $\{a \rightarrow b,\ b \rightarrow c,\ ab \rightarrow \neg c,\ a\neg b \rightarrow \neg c\}$ to generate maximally preferred models where a and c are both true.

The first thing we need to do in order to introduce $\mathcal{D}^+$ in detail is to describe what the language for representing knowledge in $\mathcal{D}^+$ is. Let $P = \{p_1, p_2, \ldots, p_n\}$ be a finite set of propositional letters, and L be the language of formulae built from P and the connectives $\neg$, $\wedge$ and $\vee$ in the standard way. We define a *default* d to be an expression of the form $\alpha \rightarrow q$, where q is a literal (i.e. a propositional letter p in P or its negation $\neg p$) and α is a set of literals[4]. We will also use the standard definition of a *model* for P as a function $M : P \mapsto \{\mathtt{True}, \mathtt{False}\}$; accordingly, we will say that M satisfies a theory T of L (written as $M \models T$) iff M assigns $\mathtt{True}$ to each formula in T, formulae in T being evaluated with respect to M in the standard manner.

The above-mentioned specificity ordering between defaults is captured by stipulating that, given a set of defaults (or *default theory*) D, a default $d \equiv \alpha \rightarrow q$ in D is *blocked* at a model M iff there exists a default d' in D such that $d' \equiv \alpha \cup \beta \rightarrow \neg q$ and $M \models \alpha \cup \beta$. A default $d \equiv \alpha \rightarrow q$ is then said to be *applicable* to a model M iff $M \models \alpha$ and d is not blocked at M. If d is applicable at M, the model $d(M)$ is defined as the model which is identical to M with the possible exception of the truth assignment to the propositional letter occurring in q, which is assigned a truth value such that $d(M) \models q$.

Naturally enough, a preference ordering induced on models by a set of defaults D may at this point be defined. Given a set of defaults D, the relation "$\leq+$" is defined to hold between models M and M' (written $M \leq+ M'$) iff there exists d in D such that d is applicable at M and such that $d(M) = M'$. The relation "$\leq$" is defined as the transitive closure of "$\leq+$"[5]. Finally,

[4]For notational convenience we will omit to draw braces in antecedents of defaults. Hence we will write e.g. $ab \rightarrow \neg c$ instead of $\{a,\ b\} \rightarrow \neg c$.

[5][MPD] defines "$\leq$" to be the *reflexive* transitive closure

we will say that a model M is *maximally preferred* (or *maximal*) with respect to a set of defaults D iff for all models M' either $M' \leq M$ is the case or $M \leq M'$ is not the case. We understand the task of reasoning in $\mathcal{D}^+$ as that of finding an arbitrary model which is maximal with respect to a given propositional theory T and a given default theory D.

We will illustrate the way $\mathcal{D}^+$ works by means of an example[6]. Let $P = \{a, b, c\}, D = \{a \rightarrow b, b \rightarrow c, ab \rightarrow \neg c, a\neg b \rightarrow \neg c\}$. **¬abc**, **¬a¬bc**, **ab¬c** and **¬a¬b¬c** are all and the only maximal models. Note that if $b \rightarrow c$ had not been blocked at **ab¬c**, then **abc** would have been maximal too, contrary to intuitions. The example is represented graphically in Figure 1.

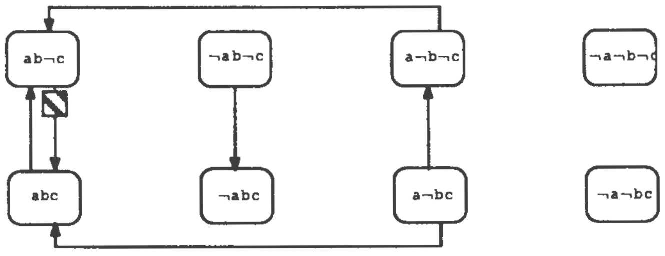


Figure 1: A simple example

3 Dealing with heterogeneous theories

In the preceding section we have described the process by which the set of maximal models is singled out from the set of all models of P through the application of a set of defaults D. It is natural, however, to require that a method be also enforced that allows a theory T of certain facts to be brought to bear in the process of maximal model selection. For instance, we might want to represent the situation in which, beside knowing that $a \rightarrow b, b \rightarrow c, ab \rightarrow \neg c$ and $a\neg b \rightarrow \neg c$, the agent also knows for sure that a is the case. In this case **¬abc**, **¬a¬bc** and **¬a¬b¬c** should no longer be maximal models, and **ab¬c** only should be endorsed. There are two methods that are described in [MPD] for bringing to bear certain knowledge in the process of maximal model selection, and their adequacy to implement a correct interaction between certain and defeasible knowledge will be a central concern of this paper. But in order to do so, we will first need to make a short digression on what we consider an "adequate" method of implementing it.

Until now we have largely proceeded on a formal level only, and are thus in need of providing some empirical justification for the formalism we have chosen. Although identifying the task of generating a vivid KB with that of finding a maximally preferred model sounds fairly intuitive, it is by no means clear why the notion of preference we have formalized should be "the right notion of preference" at all. What we need is a criterion of empirical adequacy which is independent of the formalization itself, a criterion that allows us to judge if our systems actually capture the relevant intuitions behind default reasoning as generation of a vivid KB. Different theorists might obviously have different intuitions concerning this issue; nevertheless, we will lay down our cards and describe the two minimal conditions which *we* think reasonable for a model M to be maximal with respect to a theory T and a default theory D:

1. M satisfies the theory T;
2. if M satisfies the antecedent α of a default $d \equiv \alpha \rightarrow q$ in D, then it also satisfies its consequent q, unless it also satisfies the antecedent $\alpha \cup \beta$ of a default $d' \equiv \alpha \cup \beta \rightarrow \neg q$ in D.

Someone might object that there is a lot of "tester's bias" in this criterion, and that it seems to have been laid out in order to acritically grant a stamp of adequacy to the formal definition we have presented in Section 2. We will see that this is not so, and that this criterion is demanding enough to rule out the definition of maximality enforced by $\mathcal{D}^+$. Quite immodestly, we will call a model satisfying conditions 1 and 2 an *intended model*; we will thus consider a model-preference default system empirically satisfactory iff for every set of defaults D every intended model is also a maximal model and viceversa. For instance, it is easy to check that this is indeed the case in the example of Section 2.

We may now switch back to the description of the two methods that are proposed in [MPD] in order to account for the interaction between certain and defeasible knowledge. The first of them consists in encoding certain facts by means of defaults with an empty antecedent; this style of encoding relies on the fact that the ability to reach a conclusion starting from an empty set of premises is usually taken as meaning that the conclusion is a true fact. The situation described above would then be represented by $P = \{a, b, c\}, D = \{\phi \rightarrow a, a \rightarrow b, b \rightarrow c, ab \rightarrow \neg c, a\neg b \rightarrow \neg c\}$; in this case **ab¬ c** is in fact the only maximal model, as shown in Figure 2.

But we feel that this is an unsatisfactory solution, as encoding certain facts as defaults with empty antecedents *exposes them to blockage by more specific defaults*, i.e. by items of knowledge that, although being more specific, have a weaker epistemic status than any item of certain knowledge. That this solution licenses undesired conclusions is shown by the following example. Let $P = \{a, b, c\}, D = \{a \rightarrow b, b \rightarrow c, ab \rightarrow \neg c, a\neg b \rightarrow \neg c, \phi \rightarrow a, \phi \rightarrow \neg b\}$, where the intended interpretation of the last two defaults is "a is certainly the case" and "$\neg b$ is certainly the case". The only intended model is

of "$\leq +$"; that this is redundant may be seen by inspecting the way "$\leq$" is used in the definition of maximal model.

[6]In the drawings of the following examples, rectangles will denote models represented in the obvious way (e.g. a¬bc will represent the function that assigns **True** to a and c and **False** to b). Arrows will represent "$\leq +$" relationships. A slashed arrow will represent what would have been a "$\leq +$" relationship unless a blocking had not occurred. Also, we will omit drawing arrows corresponding to self-loops (i.e. arrows starting and ending in the same model) as they do not contribute in supporting the maximality or non- of a model.

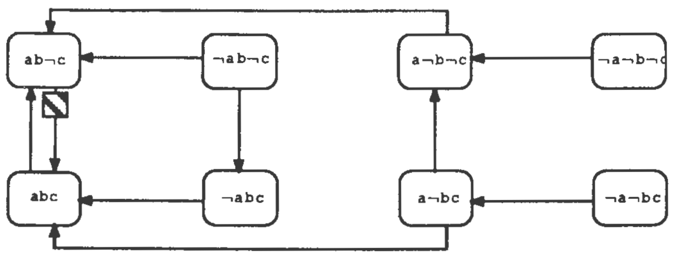


Figure 2: Defaults with empty antecedents

a¬b¬c but, as shown in Figure 3, it is not a maximal model. On the contrary, **ab¬c** is maximal (actually, it is the only maximal model) but is not intended. Basically, this happens because $a \rightarrow b$ is allowed to block $\phi \rightarrow \neg b$ at models where a is true. Instead, we think that an item of defeasible knowledge should never be allowed to invalidate an item of certain knowledge; rather, the opposite view should be enforced, with certain knowledge inhibiting the effect of defeasible knowledge when a contradiction would otherwise occur. The preceding example shows that, so to speak, this semantics (or, better, this semantics together with the style of encoding certain knowledge that it encourages) is *neither sound nor complete* with respect to the intuitive, pre-theoretical semantics of default reasoning supposedly embodied by our adequacy criterion.

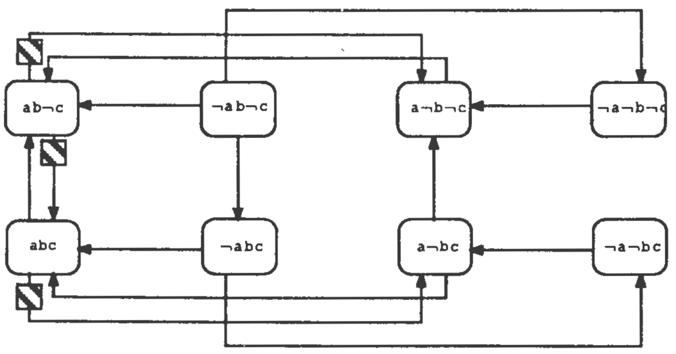


Figure 3: Improper behaviour

Let us then consider the second solution to the integration of certain and defeasible knowledge that is described in [MPD]. This solution relies on a different definition of maximality, according to which a model is maximal with respect to what we will call a *heterogeneous theory* $\langle D, T\rangle$ (where D is a default theory and T is a theory) iff $M \models T$ and for all models M' such that $M' \models T$ either $M' \leq M$ is the case, or $M \leq M'$ is not the case. In other words, in order to be maximal a model must first of all satisfy T, and its would-be maximality can only be prevented by another model which itself satisfies T. We can see that the modified system handles the preceding example correctly: let $P = \{a, b, c\}, D = \{a \rightarrow b, b \rightarrow c, ab \rightarrow \neg c, a\neg b \rightarrow \neg c\}, T = \{a, \neg b\}$. As remarked above **a¬b¬c** is the only intended model; as shown in Figure 4, it is now also the only maximal model[7].

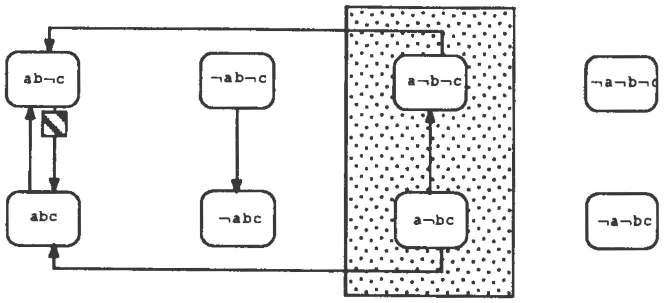


Figure 4: Maximality wrt heterogeneous theories

But we feel that this solution too is unsatisfactory, and again we feel that the reason lies in an overestimation of the role of defeasible knowledge in its interaction with certain knowledge. To see what the problems involved are, let us introduce two new definitions. We will say that there is an ***internal path*** between two models M and M' (written $M \leq^i M'$) iff there exist models $M_1, \ldots, M_n$ (with $M \equiv M_1$ and $M' \equiv M_n$) such that $M_i \leq +M_{i+1}$ for all $i = 1, \ldots, n-1$ and such that $M_i \models T$ for all $i = 1, \ldots, n$. Conversely, we will say that there is an *external path* between two models M and M' such that $M \models T$ and $M' \models T$ (written $M \leq^e M'$) iff there exist models $M_1, \ldots, M_n$ (with $M \equiv M_1$ and $M' \equiv M_n$) such that for all $i = 1, \ldots, n-1$ $M_i \leq +M_{i+1}$ and for some $j = 2, \ldots, n-1$ $M_j \not\models T$. Intuitively, an external path is a path which goes through at least one model that does not satisfy T.

Our qualms with the (second) definition of maximality described in [MDP] have to do with the fact that it still allows defeasible knowledge to override certain knowledge that contradicts it, and accomplishes this *by allowing external paths to support either the maximality or non- of a model.* We therefore proceed to give a new definition of maximality, one where external paths are ruled out from consideration, and argue that in all cases in which the original definition differs from the new one, the former yields unintuitive results while the latter does not.

The basic step of this new definition is actually the relativization of the preference ordering "$\leq+$" with respect to a theory T. We hereby define the relation "$\leq+$" wrt a heterogeneous theory $\langle D, T\rangle$ as the relation which holds between models M and M' iff $M \models T, M' \models T$ and there exists d in D such that d is applicable to M and such that $d(M) = M'$. As in the original version, "$\leq$" is defined as the transitive closure of "$\leq+$", and M is maximal wrt $\langle D, T\rangle$ iff for all M' such that $M' \models T$ either $M' \leq M$ is the case or $M \leq M'$ is not the case.

[7]In the drawings of this section grey areas represent the set of models that satisfy T.

It is simple to check that defining maximality this way is equivalent to substituting "$\leq^i$" for "$\leq$" in the original definition. In order to distinguish between the two notions of maximality we will henceforth speak of $\leq^i$-*maximality* and $\leq$-*maximality*, respectively.

In what cases $\leq^i$-maximality and $\leq$-maximality yield different results may be checked by truth table analysis. This analysis, although straightforward, is fairly tedious, and is then confined to Appendix A.. The basic result is that there are two cases in which $\leq^i$-maximality and $\leq$-maximality differ: we will call instances of the former case ***type-1 theories*** and instances of the latter ***type-2 theories***. Let us analyze them orderly.

In type-1 theories we have a model M that is not intended and is not $\leq^i$-maximal but is $\leq$-maximal. This is caused by the presence of an external path leading from M' to M and supporting the $\leq$-maximality of M, and by the absence of a similar but internal path to support its $\leq^i$-maximality. The following example is a type-1 theory. Let $P = \{a, b, c\}, D = \{ab \rightarrow c, c \rightarrow \neg a, \neg ab \rightarrow \neg c, b\neg c \rightarrow a\}, T = \{a, b\}$. abc is the only intended model and, as shown in Figure 5, is also the only $\leq^i$-maximal model. abc is also $\leq$-maximal; note, however, that also ab¬c is $\leq$-maximal, although it is not intended.

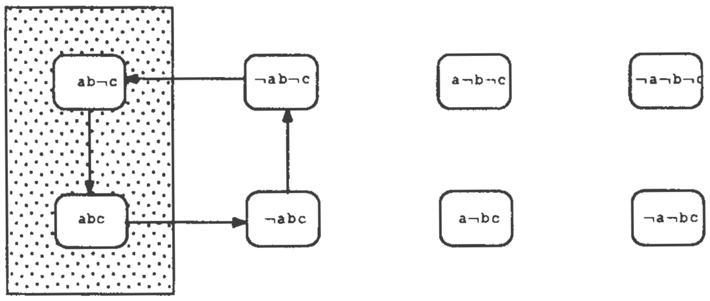


Figure 5: Improper behaviour. A type-1 theory

In type-2 theories we have quite the opposite situation, i.e. there is a model M that is intended and is $\leq^i$-maximal but is not $\leq$-maximal. This is caused by the presence of an external path leading from M to a model M' which prevents the $\leq$-maximality of M, and by the absence of a similar but internal path to prevent its $\leq^i$-maximality. The following example is a type-2 theory. Let $P = \{a, b, c\}, D = \{b\neg c \rightarrow \neg a, \neg ab \rightarrow c, bc \rightarrow a\}, T = \{a, b\}$. abc and ab¬c are the only intended models and, as shown in Figure 6, are also the only $\leq^i$-maximal models; abc is also $\leq$-maximal but ab¬c is not, although it is intended.

These examples show that also the second solution described in [MPD] to the problem of model-preference reasoning in heterogeneous default theories is, so to speak, *neither sound* (as shown by type-1 theories) *nor complete* (as shown by type-2 theories) with respect to the intuitive, pre-theoretical semantics of default reasoning. Quite apart from this, they also show that the problem lies in paths involving models that do not satisfy the theory T, and that $\leq^i$-maximality, by excluding these paths from consideration, actually implements the correct behaviour.

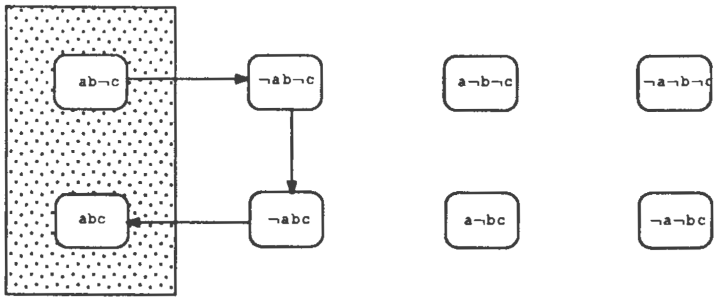


Figure 6: Improper behaviour. A type-2 theory

From an empirical standpoint, the idea that these paths and models should be neglected is informed by the general principle according to which the co-presence of knowledge endowed with a higher epistemic status (or, according to the terminology of [Gärdenfors & Makinson, 1988], knowledge that is more "epistemically entrenched") should have the reasoning process disregard knowledge endowed with a lower epistemic status that contradicts it. In our case, this means that the application of a default to a state of affairs which is inconsistent with the certain knowledge is deemed not only irrelevant but actually misleading, and so is the application of a default that yields such a state of affairs.

It might be legitimate, at this point, to ask ourselves whether $\leq^i$-maximality is sufficient in itself to make $\mathcal{D}^+$ and the other [MPD] systems comply with the adequacy criterion in Section 3. Somehow surprisingly, the answer is no. This is due to the fact that the adequacy criterion we have laid out can be met only by formalisms that make "$\rightarrow$" enjoy ***contraposition***, the property of some conditional notions "$\Rightarrow$" (e.g. material implication in classical logic, strict implication in modal logic) according to which $a \Rightarrow b$ entails $\neg b \Rightarrow \neg a$. We indeed think that the pre-theoretical notion of default implication (if only there exists such a thing) ***does*** enjoy contraposition, and this is why we did not yield to the temptation of adding "...or unless $\neg q$ belongs to T" to clause 2 of the criterion, an addition that would have exempted a formalism from endorsing contraposition. Neither the systems of [MPD] nor their modifications as resulting from $\leq^i$-maximality endorse it. Nevertheless, modifying them to this effect would be an easy matter (essentially, this would involve redefining $d(M)$ to be a set of models rather than a single model). More importantly, contraposition is not an issue which is specific to the interrelation of certain and defeasible knowledge (it is surely pertinent also to defeasible knowledge alone), and therefore falls outside the scope of this paper. We may then legitimately consider $\leq^i$-maximality as having successfully accomplished the goal of meeting the adequacy criterion.

4 Complexity issues

As one of the most important contributions of [MPD] is the establishment of complexity results for reasoning in model-preference default theories, it would be useful to understand whether these results carry over to systems where $\leq^i$-maximality is used, and whether new results can be established for such systems. As $\leq$-maximality and $\leq^i$-maximality coincide when the theory T is empty, we will be interested in results concerning the case when T is non-empty. These two cases, that we will dub the "homogeneous" and the "heterogeneous" case, respectively, have been shown in [MPD] to have, in general, different complexity. For example, the problem of reasoning in $\mathcal{DH}$ is linear in the homogeneous case, while in the heterogeneous case it is polynomial if T is a set of literals and NP-hard if T is a set of Horn clauses. Also, for some of these systems results are known for the homogeneous case but not for the heterogeneous case; in particular, for the homogeneous case NP-hardness results for $\mathcal{D}$ and $\mathcal{DH}^+$ and a polynomial result for $\mathcal{DH}^+_a$ are reported in [MPD], but the corresponding results for the heterogeneous case are still unknown.

As remarked in [MPD], the problem of finding a model which is maximal wrt a pair $\langle D, T\rangle$ is best viewed as a search problem [Garey & Johnson, 1979]. The main result of this paper is that, for any model-preference default system considered in [MPD] and variations thereof, if T is a set of literals and $\leq^i$-maximality is used, the search problem in the heterogeneous case is no harder than the homogeneous case. The result is based on the key observation that, in both cases, only internal paths are allowed (internal to the set of all models of the language P in the former case, and to the set of all models satisfying T in the latter), whereas, according to $\leq$-maximality, external paths are allowed in the heterogeneous case but not in the homogeneous case.

Theorem. The search problem for $\mathcal{S}^i$ with HD-theories $\langle D, T\rangle$ (with T a set of literals) belongs to the same complexity class of the search problem for $\mathcal{S}$ with empty T, where $\mathcal{S}$ is any model-preference default system discussed in [MPD] and $\mathcal{S}^i$ is the version of $\mathcal{S}$ relying on $\leq^i$-maximality. □

In [Sebastiani, 1990b] we prove this theorem by giving an algorithm that, given an HD-theory $\langle D, T\rangle$ (where T is a set of literals) defined on some alphabet P, builds a default theory D' defined on $P - P_T$ (where P_T is the set of propositional letters in P that occur in T) such that, whenever a model M is $\leq$-maximal wrt D', the model $M \cup T$ is $\leq^i$-maximal wrt $\langle D, T\rangle$[8]. This algorithm is linear in the number of occurrences of literals in D. Essentially, what the algorithm does is inspecting the set D, eliminating from it any default that does not contribute to the determination of $\leq^i$-maximality (either because its consequent is in T, or because its antecedent or its consequent are inconsistent with T) and eliminating from the antecedents of the remaining defaults all literals that already appear in T. The models of P that satisfy T and the preference relations between them that are induced by D' form a graph that is isomorphic to the one formed by the models of P–P_T and by the preference relations between them that are induced by D'. Due to the isomorphism, there is a one-to-one correspondence between $\leq^i$-maximal models endorsed by the former graph and $\leq$-maximal models endorsed by the latter.

The following table summarizes complexity results in the heterogeneous case (with T a set of literals) for the main model-preference default systems.

	$\mathcal{D}$	$\mathcal{DH}$	$\mathcal{DH}^+$	$\mathcal{DH}^+_a$
$\leq$	?	P	?	?
$\leq^i$	$NP-hard$	*linear*	$NP-hard$	P

Also, note that although the homogeneous and heterogeneous case have the same theoretical complexity, the heterogeneous case is in practice much simpler (somehow in contrast with what happens for $\leq$-maximality). For example, the search problem for the homogeneous case of $\mathcal{DH}$ is $\mathcal{O}(n)$, where n is the number of occurrences of literals in D. Although the heterogeneous case (with $\leq^i$-maximality) of $\mathcal{DH}$ is still $\mathcal{O}(n)$, n is now the the number of occurrences of literals in D', a subset of D. That by adopting $\leq^i$-maximality the heterogeneous case should be simpler is also apparent from the fact that the presence of a theory T consisting of a set of k literals transforms the problem of searching the set of all models of the propositional language P into the one of searching only the set of those models that satisfy T: the latter graph has 2^k times less nodes than the former. This accounts for the rather intuitive fact that, as the amount of certain information increases, the amount of computational resources required to "flesh out" the knowledge base should proportionally decrease.

5 Conclusion

We have shown how two methods proposed in [MPD] fail, for different reasons, to capture what we think the correct interaction between certain and defeasible information should be. The modified definition of maximality ("$\leq^i$-maximality") we have provided has been shown to be the notion of maximality that better represents our intuitions of how default reasoning should accommodate the interaction between these two fundamentally different types of information. When this notion of maximality is plugged in, not only reasoning in the presence of a theory T consisting of a set of literals is provably no harder than reasoning with an empty T, but is likely to be, in practice, much more efficient.

Acknowledgements

This research was carried out in the framework of the TELOS project, dealing with the design and construction of a knowledge representation language supporting the phase of requirements modelling in information systems development. I would like to thank John Mylopoulos for encouraging the work described here and for pro-

[8]For better convenience we here think of a model as the largest set of literals that is satisfied by it.

viding me the environment where this research was carried out. I would also like to thank David Lauzon and Bart Selman for our stimulating discussions about the topic of this paper.

A Comparing $\leq^i$-maximality and $\leq$-maximality

The truth table that establishes type-1 and type-2 theories as all and the only types of theories in which $\leq^i$-maximality and $\leq$-maximality differ is spelled out in Table 1. The $\leq^i$-maximality (column v) and $\leq$-maximality (column vi) of a model M satisfying a propositional theory T is represented as a function of the simple conditions $\forall M'.(M' \leq^i M)$ (column i), $\exists M'.(\neg M' \leq^i M \wedge M \leq^i M')$ (column ii), $\forall M'.(M' \leq M)$ (column iii) and $\exists M'.(\neg M' \leq M \wedge M \leq M')$ (column iv); all models considered in this Appendix satisfy T. Condition 5 is the conjunction of 1 and the negation of 2, while 6 is the conjunction of 3 and the negation of 4.

Note that 9 of the 16 rows are not even taken into consideration as, for different reasons, they depict impossible situations. For instance, rows 1 through 4 are ruled out on the grounds that the conditions corresponding to columns i and ii cannot be true at the same time, unlike the simultaneous presence of **True** in the first two columns of these rows would imply. Similarly, the simultaneous presence of **True** in columns iii and iv rules out rows 1, 5, 9 and 13; the simultaneous presence of **True** in columns i and iv rules out rows 1, 3, 5 and 7, and the simultaneous presence of **True** in column i and of **False** in column iii rules out rows 3, 4, 7 and 8. In all these cases, the incompatibility of the two conditions corresponding to the two columns at issue is easy to check.

Among the 7 rows that are left, rows 6, 11, 14 and 16 agree on the values assigned to $\leq^i$-maximality and $\leq$-maximality. Instead, rows 10, 12 and 15 assign different values to the two notions.

However, a closer analysis shows that the truth conditions corresponding to rows 10 and 12 actually come down to representing the same situation, as far as the motivations for the dissimilarity between $\leq^i$-maximality and $\leq$-maximality are concerned. In fact, row 10 corresponds to the situation in which $\exists M'.(\neg M' \leq^i M \wedge M \leq^i M')) \wedge \forall M'.(M' \leq M)$ while row 12 corresponds to the situation in which $\exists M'.(\neg M' \leq^i M \wedge M \leq^i M') \wedge \exists M''.(\neg M'' \leq M) \wedge \forall M'''.(M''' \leq M \wedge \neg M \leq M'''))$; the only difference is that in row 10 the existence of an "isolated model" (i.e. one which has no relation of preference whatsoever, either outgoing or incoming, with M) is stated, while the existence of such a model is ruled out in row 10. Since the mere existence of such a model does not have any influence in supporting the truth or falsity of the conditions corresponding to columns v and vi, we may consider the situations corresponding to rows 10 and 12 to be actually the same situation.

Row 10 and 12 correspond then to what we have called type-1 theories, while row 15 corresponds to type-2 theories.

A.1 Table 1

	i	ii	iii	iv	v	vi
1	**True**	**True**	**True**	**True**		
2	**True**	**True**	**True**	**False**		
3	**True**	**True**	**False**	**True**		
4	**True**	**True**	**False**	**False**		
5	**True**	**False**	**True**	**True**		
6	**True**	**False**	**True**	**False**	**True**	**True**
7	**True**	**False**	**False**	**True**		
8	**True**	**False**	**False**	**False**		
9	**False**	**True**	**True**	**True**		
10	**False**	**True**	**True**	**False**	**False**	**True**
11	**False**	**True**	**False**	**True**	**False**	**False**
12	**False**	**True**	**False**	**False**	**False**	**True**
13	**False**	**False**	**True**	**True**		
14	**False**	**False**	**True**	**False**	**True**	**True**
15	**False**	**False**	**False**	**True**	**True**	**False**
16	**False**	**False**	**False**	**False**	**True**	**True**

References

[Etherington *et al.* , 1989] Etherington, David W.; Borgida, Alex; Brachman, Ronald J. & Kautz, Henry A. Vivid knowledge and tractable reasoning: preliminary report. In *Proceedings of IJCAI-89*, Detroit, MI, pp. 1146–1152, 1989.

[Gärdenfors & Makinson, 1988] Gärdenfors, Peter & Makinson, David. Revisions of knowledge systems using epistemic entrenchment. In *Proceedings of the 2nd Conference on Theoretical Aspects of Reasoning about Knowledge*, Monterey CA, pp. 83–95, 1988.

[Garey & Johnson, 1979] Garey, Michael R. and Johnson, David S. *Computers and intractability.* New York NY: Freeman, 1979.

[Johnson-Laird, 1983] Johnson-Laird, Philip N. *Mental models.* Cambridge, MA: Harvard University Press, 1983.

[Konolige, 1987] Konolige, Kurt. On the relation between default theories and autoepistemic logic. In *Proceedings of IJCAI-87*, Milan, Italy, pp. 394–401. [1] An extended version appears as "On the relation between default logic and autoepistemic theories" in *Artificial Intelligence 35*, pp. 343–382, 1987.

[Levesque, 1986] Levesque, Hector J. Making believers out of computers. *Artificial Intelligence 30*, pp. 81–108, 1986.

[Levesque, 1988] Levesque, Hector J. Logic and the complexity of reasoning. *Journal of Philosophical Logic 17*, pp. 355–389, 1988.

[McCarthy, 1980] McCarthy, John. Circumscription - A form of nonmonotonic reasoning. *Artificial Intelligence 13*, pp. 81–108, 1980. [1] Appears also in Ginsberg, Matthew L. (ed.) (1987), *Readings in nonmonotonic reasoning*, Los Altos, CA: Morgan Kaufmann, pp. 145–152.

[McCarthy, 1986] McCarthy, John. Applications of circumscription to formalizing commonsense knowledge.

Artificial Intelligence 28, pp. 89–116, 1986. [1] Appears also in Ginsberg, Matthew L. (ed.) (1987), *Readings in nonmonotonic reasoning*, Los Altos, CA: Morgan Kaufmann, pp. 153–166.

[Moore, 1985] Moore, Robert C. Semantical considerations on nonmonotonic logic. *Artificial Intelligence 25*, pp. 75–94, 1985. [1] Appears also in Ginsberg, Matthew L. (ed.) (1987), *Readings in nonmonotonic reasoning*, Los Altos, CA: Morgan Kaufmann, pp. 127–136.

[Sebastiani, 1990a] Sebastiani, Fabrizio. A fully denotational semantics for model-preference default systems. Technical Report IEI-B4-06-1990, Istituto di Elaborazione dell'Informazione - CNR, Pisa, Italy, 1990.

[Sebastiani, 1990b] Sebastiani, Fabrizio. On heterogeneous model-preference default theories (extended version). Forthcoming Technical Report, Istituto di Elaborazione dell'Informazione - CNR, Pisa, Italy, 1990.

[Selman & Kautz, 1988] Selman, Bart & Kautz, Henry A. The complexity of model-preference default theories. In *Proceedings of the 1988 Conference of the Canadian Society for Computational Studies of Intelligence*, Edmonton, Alberta, pp. 102–109, 1988. [1] Appears also in Reinfrank, Michael; De Kleer, Johan; Ginsberg, Matthew L. & Sandewall, Erik (eds.) (1989), *Nonmonotonic reasoning*, Heidelberg, BRD: Springer, pp. 115–130. [2] An extended version is forthcoming as "Model-preference default theories" in *Artificial Intelligence*.

The Complexity of Horn Theories with Normal Unary Defaults

Jonathan Stillman
Artificial Intelligence Program
General Electric Research and Development Center
P.O. Box 8
Schenectady, N.Y. 12301
e-mail: stillman@crd.ge.com

Abstract

We solve an open problem stated in [Kautz and Selman, 1989], showing that although fast algorithms exist for determining whether a literal holds in a propositional default theory in which the propositional theory consists solely of literals and the default rules are *Horn* (see [Kautz and Selman, 1989]), and exist for deciding satisfiability of propositional Horn theories, the two cannot be combined without introducing intractability. In particular, we show that when the propositional theory of a default theory allows Horn clauses, the membership problem becomes intractable even when the default rules in the theory are restricted to being propositional *normal unary* default rules, a strong restriction of propositional Horn default rules.

We also present several related results, showing that the entailment problem, the enumeration problem, and the problem of determining whether there exists an extension that "satisfies" some specified number of the default rules are all intractable for these restricted default theories.

1 Introduction

One of the central concerns of artificial intelligence research involves developing useful models of how one might emulate on computers the 'common-sense' reasoning in the presence of incomplete information that people do as a matter of course. Traditional predicate logics, developed for reasoning about mathematics, are inadequate as a formal framework for such research in that they are inherently *monotonic*: if one can derive a conclusion from a set of formulae then that same conclusion can also be derived from every superset of those formulae. It is argued that people simply don't reason this way: we are constantly making assumptions about the world and revising those assumptions as we obtain more information (see [McCarthy, 1977] or [Minsky, 1975], for instance).

Many researchers have proposed modifications of traditional logic to model the ability to revise conclusions in the presence of additional information (see, for instance, [McCarthy, 1986], [Moore, 1983], [Poole, 1986]). Such logics are called *nonmonotonic*. Informally, the common idea in all these approaches is that one may want to be able to "jump to conclusions" that might have to be retracted later. While a detailed discussion of nonmonotonic logics is outside the scope of this paper, a good introduction to the topic can be found in [Etherington, 1988], and a number of the most important papers in the field have been collected in [Ginsberg, 1987].

One of the most prominent of the formal approaches to nonmonotonic reasoning, developed by Reiter ([Reiter, 1980]), is based on *default rules*, which are used to model decisions made in prototypical situations when specific or complete information is lacking. Reiter's *default logic* is an extension of first order logic that allows the specification of default rules, which we will summarize shortly. Unfortunately, the decision problem for Reiter's default logic is highly intractable in that it relies heavily on consistency checking for processing default rules, and is thus not even semi-decidable (this is not a weakness of Reiter's logic alone; it is common to most nonmonotonic logics). This precludes the practical use of Reiter's default logic in most situations.

The motivation for searching for computationally tractable inference mechanisms for subclasses of *propositional* default reasoning is based on the need to reason about relatively large propositional knowledge bases in which the default structures may be quite simple. Recent research involving inheritance networks with exceptions is particularly relevant, and is explored in depth in [Touretzky, 1986] and in Chapter 4 of [Etherington, 1988], where the close relationship between default logic and inheritance networks with exceptions is explored.

In order to gain computational tractability of reasoning in default logic, one must restrict expressiveness considerably. If one simply restricts the logic to reasoning about arbitrary propositions, the resulting decision problems are at least as hard as deciding standard propositional logic, regardless of any restrictions on the types of default rules allowed. Since the satisfiability problem is intractable for propositional logic, one must consider further restrictions.

Recently, Kautz and Selman ([Kautz and Selman, 1989]) investigated a number of restricted default logics defined over subsets of propositional calculus with various restrictions on the syntactic form of default rules allowed. They described a partial order of such restrictions, and analyzed the complexity of several problems over this partial order when the propositional theory is restricted to a set of literals. Several restrictions on the syntactic form of default rules were shown to result in polynomial-time tests for determining whether certain properties hold given such a restricted propositional theory. In particular, it was shown that one can decide in polynomial time whether there exists an *extension* that contains a given literal when the default rules are restricted to a class they called *Horn* default rules. They suggested that the ability to combine such default theories with non-default propositional Horn theories would be particularly useful, but left open the question of whether the membership problem (i.e., determining whether there exists an extension of a given default theory containing a specified literal) for such a combination of theories is tractable. One of the main theorems of this paper shows that a strong restriction of this problem is NP-complete.

The remainder of this paper is organized as follows: we begin with a brief description of Reiter's default logic, followed by a short overview of NP-completeness, and a presentation of the restrictions considered by Kautz and Selman. In Section 3 we prove that it is NP-complete to determine whether a default theory consisting of non-default propositional Horn clauses together with normal unary default rules contains a given literal. In Section 4, we discuss several related results. Finally, we summarize the results presented and discuss areas for further research.

2 Preliminaries

2.1 Reiter's Default Logic

For a detailed discussion of Reiter's default logic the interested reader is referred to [Reiter, 1980]. In this section we will simply review some of the immediately pertinent ideas.

A *default theory* is a pair (D, W), where W is a set of closed well-formed formulae (wffs) in a first order language and D is a set of *default rules*. A *default rule* consists of a triple $< \alpha, \beta, \gamma >$, where

α is a formula called the *prerequisite*,

β is a set of formulae called the *justifications*, and

γ is a formula called the *conclusion*.

Informally, a default rule denotes the statement "if the *prerequisite* is true, and the *justifications* are consistent with what is believed, then one may infer the *conclusion*."

Default rules are written

$$\frac{\alpha : \beta}{\gamma}$$

If the conclusion of a default rule occurs in the justifications, the default rule is said to be *semi-normal*; if the conclusion is identical to the justifications the rule is said to be *normal*.

A default rule is *closed* if it does not have any free occurrences of variables, and a default theory is *closed* if all of its rules are closed.

The maximally consistent sets that can follow from a default theory are called *extensions*. An extension can be thought of informally as one way of "filling in the gaps about the world."

Formally, an extension E of a closed set of wffs T is defined as the fixpoint of an operator Γ, where $\Gamma(T)$ is the smallest set satisfying:

$W \subseteq \Gamma(T)$,

$\Gamma(T)$ is deductively closed,

for each default rule $d \in D$, if the *prerequisite* is in $\Gamma(T)$, and T does *not* contain the negations of any of the *justifications*, then the *conclusion* is in $\Gamma(T)$.

Since the operator Γ is not necessarily monotonic, a default theory may not have any extensions. Normal default theories do not suffer from this, however (see [Reiter, 1980]), and always have at least one extension.

There are several important properties that may hold for a default theory. Given a default theory (D, W), perhaps together with a literal q, one might want to determine the following about its extensions:

Existence Does there exist *any* extension of (D, W)?

Membership Does there exist an extension of (D, W) that contains q? (This is called *goal-directed reasoning* by Kautz and Selman.)

Entailment Does every extension of (D, W) contain q? (This is closely related to *skeptical reasoning*, where a literal is believed if and only if it is included in all extensions.)

2.2 NP-complete Problems

NP is defined to be the class of languages accepted by a nondeterministic Turing machine in time polynomial in the size of the input string. An important subset of NP is the class P, the class of languages accepted by a *deterministic* Turing machine in polynomial time. These problems* comprise those we usually consider *tractable*, in that the time needed to solve them is polynomially related to the problem size.

The "hardest" languages in NP are called NP-complete: NP-complete languages share the property that all languages in NP can be transformed into them via some polynomial time transformation. To show that a problem in NP is NP-complete one must demonstrate a polynomial-time transformation of an instance of a known NP-complete problem to an instance of the problem under consideration in such a way that a solution to one indicates a solution to the other. The known NP-complete problem we will use in this paper is called 3SAT, and is stated formally as follows:

3-SATISFIABILITY (3SAT)

Instance: A finite set $C = \{c_1, \ldots, c_m\}$ of propositional clauses, each of which consists of exactly 3 *literals* (propositional variables or their negations).

Question: Does there exist a truth assignment that satisfies C?

*NP-completeness is often discussed in terms of *decision problems* rather than languages, although the two are interchangeable.

The theory of NP-completeness is relatively well-understood; for a thorough and readable discussion of the topic the interested reader is referred to [Garey and Johnson, 1979]. The fastest known deterministic algorithms for NP-complete problems take time exponential in the problem size. It is not known whether this is necessary: one of the central open problems in computer science is whether P = NP. Most researchers believe that P $\neq$ NP, and that NP-complete problems really do need exponential time to solve. Thus these problems are considered *intractable*, since if P $\neq$ NP, we cannot hope to solve instances of them with inputs of nontrivial size.

Demonstrating the NP-completeness of a problem does not necessarily imply that it it cannot be solved in practice: sometimes (e.g., the Traveling Salesman Problem) good polynomial approximation algorithms have been devised. Unfortunately, it is not clear what might comprise a reasonable *approximation* to an extension in a default theory. Even when approximation algorithms do not apply, there are often important subclasses of hard problems that can be solved efficiently (deciding satisfiability of propositional Horn clauses is a good example of such a situation). Alternatively, perhaps many of the instances that may arise in practice will have structural properties that can be used to gain tractability. Knowing that a problem is NP-complete is important, however, in that it suggests that exact solutions are unlikely to be obtainable for nontrivial instances, and that some additional restrictions may need to be made on the structure of the problem being considered.

2.3 A Taxonomy of Default Theories

In [Kautz and Selman, 1989], Kautz and Selman presented a taxonomy of propositional default theories. They restricted W to contain only propositional literals (i.e., propositional variables and their negations), and restricted default rules to be semi-normal rules in which the precondition, justifications, and conclusions of each default rule consisted of conjunctions of literals (this restriction makes consistency checking a simple task). They also considered the following further restrictions on the default rules allowed.

Unary The prerequisite of each default rule must be a positive literal, and the conclusion must be a literal. If the consequence is positive, the justification must be the conjunction of the consequence and a single negative literal; otherwise, the justification must be the consequence.

Disjunction-Free Ordered The reader is referred to [Etherington, 1988] for a formal definition of ordered theories; intuitively, in ordered theories the literals can be ordered in such a way that potentially unresolvable circular dependencies cannot occur.

Ordered Unary These combine the restrictions of the first two theories described above. Kautz and Selman remark that these theories appear to be the simplest necessary to represent inheritance hierarchies with exceptions ([Touretzky, 1986]).

Disjunction-Free Normal These are disjunction-free ordered theories in which the consequence of each default rule is identical to the justification.

Horn The prerequisite literals in these default rules must each be positive, and the justification and consequence are identical, each consisting of a single literal.

Normal Unary The prerequisite in each of these default rules consists of a single positive literal, the conclusion must be a literal, and the justification must be identical to the consequence. These form the most simple class of default rule that is considered in [Kautz and Selman, 1989].

These restricted theories are related in a partial order as shown in Figure 1 below.

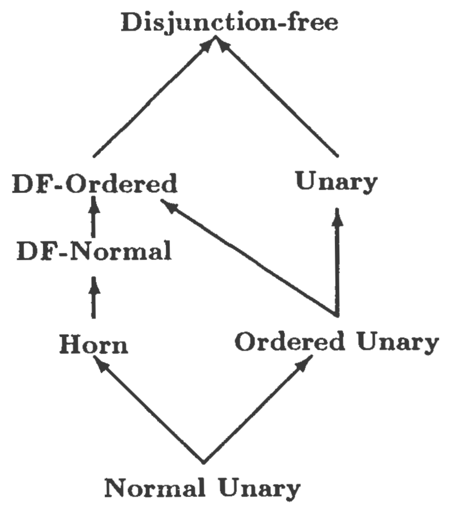


Figure 1: Kautz and Selman's hierarchy of restricted default theories.

3 Main Results

Quite often, a default theory will have multiple extensions, and one may want to restrict examination to a limited number of them. One important measure of which extensions to consider may be the inclusion of some particular propositions. As mentioned above, this is variously referred to as *goal-directed reasoning* and the *membership problem*. Figure 2 summarizes Kautz and Selman's results with regard to the taxonomy they described. In particular, it is shown that for the class of Horn default theories, goal-directed reasoning can be done in linear time when the propositional theory consists of propositional literals. They suggest that although this is somewhat useful, it would be much more interesting if one could combine such default rules with propositional Horn theories efficiently. More formally, one would like to solve the following problem:

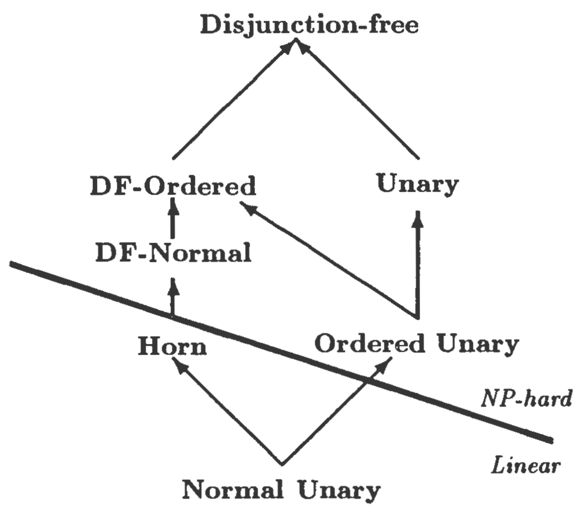


Figure 2: The complexity of goal-directed reasoning in the restricted default theories considered by Kautz and Selman.

Horn Clauses with Normal Unary Defaults

Instance: A finite set H of propositional Horn clauses, together with a finite set D of normal, unary, propositional default rules, and a distinguished literal q.

Question: Does there exist an extension of (D, H) that contains the literal q ?

In this section we show that this problem is intractable, proving:

Theorem 1 Horn Clauses with Normal Unary Defaults *is NP-complete.*

Proof: It is not difficult to demonstrate membership in NP: although the extension may be too large to describe explicitly, it suffices to provide the original set of Horn clauses, together with those default rules that were applied, and verify that the default rules form a maximal set that can actually be applied consistently. Since these are disjunction-free, this can be done efficiently.

To demonstrate NP-hardness we transform an instance of 3SAT to one of Horn Clauses with Normal Unary Defaults as follows. Given an instance I of 3SAT, we begin by converting I into a new set of clauses consisting of a set H of Horn clauses together with a set P of clauses each of which contain exactly two literals, each occurring positively. To do this we simply place each clause in I that contains at most one positive literal into H; the remaining clauses contain either two or three positive literals. For each of the remaining clauses, choose one of the positive literals (call it x), introduce a new variable $\hat{x}$ and the clauses

$$(\neg x \vee \neg \hat{x}),$$

which is a Horn clause and is placed into H, and

$$(x \vee \hat{x}),$$

which is placed into P. These two clauses taken together are the clausal form of the formula

$$(x \Leftrightarrow \neg \hat{x}).$$

Finally, replace the occurrence of x in the original clause with $\neg\hat{x}$. The resulting clause has one less positive literal than the original; if it is now a Horn clause, place it in H. Otherwise repeat the replacement process to remove one of the remaining positive literals. Note that since equivalence of each literal x and the new corresponding literal $\neg\hat{x}$ is enforced by the added clauses, every satisfying assignment for the original formula can be extended easily to a satisfying assignment for the new formula, and vice versa. The transformation has the property, however, that there are more falsifying assignments for the new formula than for the original. Note also that this transformation only results in a linear increase in the size of the problem.

At this point, we have a set H of Horn clauses, which, together with one more clause we will add later, will comprise the propositional part of the default theory we are constructing. Since the clauses in P are non-Horn, they cannot be included in the propositional part of the theory. Thus, we must construct a set of normal unary default rules D to model the clauses in P. This is done as follows.

For each variable a that appears in some clause in P, we introduce the default rule

$$\frac{: a}{a}$$

into D. Let us assume that P contains m clauses, i.e., $P = \{c_1, \ldots, c_m\}$. Each of these is of the form $c_i = (a \vee b)$, where a and b are positive literals. For each such clause, introduce a new propositional variable q_i, and introduce the following default rules into D:

$$i_1.\ \frac{a : q_i}{q_i} \qquad i_2.\ \frac{b : q_i}{q_i} \qquad i_3.\ \frac{: \neg q_i}{\neg q_i}$$

Once this is done for each of the clauses in P, we introduce one additional new variable and a final Horn clause into H to complete the construction:

$$H_q = (\neg q_1 \vee \neg q_2 \vee \ldots \vee \neg q_m \vee q)$$

This phase of the transformation also results in at most linear growth in problem size. We now show that there exists an extension of (D, H) that contains q if and only if the original formula F is satisfiable.

($\Rightarrow$). Suppose F is satisfiable. Since we replaced the clauses in P with a set of default rules, we must show that we can, given a satisfying assignment α for F, construct an extension of (D, H) that contains q. It is easy to see that α can be extended to an assignment α' in which those new variables introduced in transforming F to the sets H and P are assigned truth values so that all the clauses in $H \cup P$ are satisfied, and in fact, that the assignment of values to the new variables is completely determined by α. We use this assignment as the basis for the extension we will construct. We proceed as follows. Each of the clauses in P must have had one of its variables assigned the value *true* by α'. For each of these clauses $c_i = (a \vee b)$ we observe that if a is assigned the value *true* by α', we can apply the default rules

$$\frac{: a}{a} \qquad \frac{a : q_i}{q_i}$$

We can proceed similarly if b is assigned the value *true*. Note that since there are no propositional constraints on the variables q_i other than the single Horn clause we added, we can always consistently add these. When this has been done for each of the clauses, it follows from the Horn clause H_q that the set we have specified also contains q. It is a straightforward matter to confirm that this set can be augmented via deductive closure to form an extension of (D, H) that includes q, since no other default rules can be applied, and the only new Horn clause added (H_q) is also satisfied.

($\Leftarrow$). Suppose that there exists an extension of (D, H) that includes q. Since H contains only non-unit Horn clauses, it is easily seen to be consistent, thus it has only consistent extensions (see [Reiter, 1980]). Thus we need only show that each formula from P can be satisfied consistently with H. Since we are given that q is contained in the extension, we can infer from the clause H_q that each of the $\{q_i | 1 \leq i \leq m\}$ are also in the extension (otherwise the extension would contain q_i for some $1 \leq i \leq m$, and the clause H_q would be satisfied without forcing q to be true). For an arbitrary clause $c_i = (a \vee b)$ from P, the default rules

$$\frac{a : q_i}{q_i} \qquad \frac{b : q_i}{q_i}$$

are the only default rules that could have admitted q_i into the extension. The prerequisites of these default rules ensure that at least one of a, b is also in the extension (they may have been included using the default rules

$$\frac{: a}{a} \qquad \frac{: b}{b}$$

or as a consequence of including other literals). Thus, for each of the clauses in P at least one of its literals is in the extension. Since this extension is consistent with H, the set $P \cup H$ is also consistent, and the theorem follows. □

4 Related Results

In this section we present several results that can be obtained by making minor modifications to the proof presented above.

Theorem 2 *It is co-NP-complete to determine whether a specified literal q holds in every extension of a default theory (D, H), where H is a finite set of propositional Horn clauses, and D is a finite set of normal, unary, propositional default rules.*

Proof (sketch): The transformation above is modified by adding a default rule

$$\frac{: \neg q}{\neg q}$$

to D, causing the literal $\neg q$ to be included explicitly in every extension if and only if the original instance of 3SAT is unsatisfiable, and the result follows. □

Theorem 3 *It is #-P-complete to count those extensions of a default theory (D, H) containing a specified literal q, where H is a finite set of propositional Horn clauses, and D is a finite set of normal, unary, propositional default rules.*

Proof (sketch): We modify the original transformation by adding default rules corresponding to *each* of the original variables and their negations, rather than just those in P, thus eliminating "don't care" situations that might otherwise arise in extensions, in which for some propositional variables neither the variable or its negation are in the extension. This modified transformation induces a situation in which each extension containing the specified literal q corresponds to a unique satisfying truth assignment for the original formula, and vice versa. The result follows immediately. □

The problems addressed in Theorem 2 and Theorem 3 are closely related to *skeptical reasoning* discussed in [Touretzky, 1986]. A skeptical reasoning system accepts a proposition only if it is included in *every* extension. It was shown in [Kautz and Selman, 1989] that normal unary default theories have an $O(n^3)$ algorithm for determining whether a proposition holds in all extensions. Theorem 2 demonstrates that if one extends the theory to allow Horn clauses in the non-default part, such skeptical reasoning becomes intractable. Theorem 3 shows that for such default theories it is also intractable to determine whether a proposition holds in *most* extensions. As a result, even approaches approximating skeptical reasoning by accepting propositions that are included in *most* extensions are intractable in these theories.

It is also interesting to note that the construction we describe has the property that for each clause appearing in P, exactly one of the literals in that clause will be true in a given satisfying assignment. The next theorem shows that even determining whether there is an extension that "satisfies" a given number of one's default rules is NP-complete. Since default rules are often used to express descriptions of "preferred interpretations," such queries provide an indication of how close one might be able to get to one's preferences.

Theorem 4 *It is NP-complete to determine, given a default theory (D, H), where H is a finite set of propositional Horn clauses, and D is a finite set of normal, unary, propositional default rules with empty prerequisites and with positive justifications and conclusions*[†] *and a positive number k, whether there is an extension of (D, H) that contains the consequences of at least k of the default rules in D.*

Proof: The construction of H and P is exactly as in the proof of Theorem 1 above. Note that for each clause $(a \vee b)$ in P there is a corresponding clause $(\neg a \vee \neg b)$ in H. This forces *exactly* one of a and b to be true in any satisfying assignment for $H \cup P$. In order to make sure that applying a default rule corresponds to satisfying *exactly* one clause from P, we must ensure that no variable

[†]These form the most simple possible type of default rule, expressing the desire to believe some propositional variable whenever it is consistent to do so.

appears in more than one clause in P. To do this, we proceed as follows. If a variable a appears in two clauses in P, introduce a new variable a', Horn clauses

$$(\neg a \vee a')$$

and

$$(\neg a' \vee a),$$

and replace one occurrence of a in P by a'. When this process is completed, each variable appearing in P appears exactly once in P. Next, for each literal a appearing in P add the default rule

$$\frac{:a}{a}$$

Let m be the number of clauses in P. If the original formula is satisfiable then we can easily extend this to an extension in which exactly m of the default rules were applied, since exactly one of the variables in each clause from P can be true. Similarly, since it is inconsistent for an extension to contain both variables from any clause in P, if there is an extension in which exactly m default rules were applied, exactly one variable from each clause in P is true. Since the clauses in H are consistent, the entire formula is satisfiable. □

5 Discussion

We have shown that several problems associated with restricted propositional default theories are intractable, despite the fact that there exist tractable algorithms for their component parts. These default theories are quite simple, and our results show that unless P = NP, in order to effectively reason in default theories one must live with constraints that are quite limiting, some of which are described in [Kautz and Selman, 1989].

The most promising area for further study involves identifying different restrictions that yield tractable reasoning methods without sacrificing expressibility to the point where only trivial default theories can be reasoned about. We are currently investigating several possibilities, and will present a number of new results related to the problem of reasoning in restricted propositional default theories in a forthcoming paper.

Acknowledgements

The author is indebted to Deepak Kapur, Robert Mattheyses, and to the referees for helpful comments on earlier versions of this work.

References

[Etherington, 1988] David W. Etherington. *Reasoning with Incomplete Information.* Pitman, London, 1988.

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability.* W.H. Freeman, 1979.

[Ginsberg, 1987] Matthew L. Ginsberg, editor. *Readings in Nonmonotonic Reasoning.* Morgan Kaufman, Los Altos, CA, 1987.

[Kautz and Selman, 1989] Henry A. Kautz and Bart Selman. Hard problems for simple default logics. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 189–197, Toronto, Ontario, Canada, 1989.

[McCarthy, 1977] John McCarthy. Epistemological problems of artificial intelligence. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 1038–1044. International Joint Committee on Artificial Intelligence, 1977.

[McCarthy, 1986] John McCarthy. Applications of circumscription to formalizing commonsense knowledge. *Artificial Intelligence*, 28:89–166, 1986.

[Minsky, 1975] Marvin Minsky. A framework for representing knowledge. In Patrick Winston, editor, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill, New York, 1975.

[Moore, 1983] Robert C. Moore. Semantical considerations on nonmonotonic logic. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 272–279, Karlsruhe, West Germany, 1983. International Joint Committee on Artificial Intelligence.

[Poole, 1986] David L. Poole. Default reasoning and diagnosis as theory formation. Technical Report CS-86-08, Dept. of Computer Science, University of Waterloo, 1986.

[Reiter, 1980] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Touretzky, 1986] David S. Touretzky. *The Mathematics of Inheritance Systems.* Pitman, London, 1986.

Student Model Revision: Evolution and Revolution

Xueming Huang, Gordon I. McCalla and Jim E. Greer

ARIES Laboratory
Department of Computational Science
University of Saskatchewan
Saskatoon, Saskatchewan, Canada S7N 0W0

Abstract

To build a long-term, individual student model, two difficult issues must be dealt with: belief revision and reasoning with incomplete knowledge. This paper presents a Student Model Maintenance System (SMMS) which allows two types of revision in a student model: evolutionary revision and revolutionary revision. It also ties the two types of revision together by allowing an evolutionary revision to trigger a revolutionary revision. In the SMMS, a Default Package Network (DPN) is constructed to represent student knowledge assumed by the tutor in the absence of full information. These research results apply not only to student models, but also to other user models.

1. Introduction

Intelligent systems that interact with people need knowledge about the user. This kind of knowledge is usually referred to as the system's *user model. Student models* are the user models of intelligent tutoring systems (ITS's) [Sleeman and Brown, 1982] [Wenger, 87] [Clancey, 1986]. User models can be categorized in three dimensions [Rich, 1979]: canonical or individual models; short-term or long-term models; explicitly told (by the user) or implicitly abstracted (by the system) models. This paper concerns individual, long-term student models.

One prominent property of student models is their non-monotonicity, that is, some information in the model may be retracted after interaction with the student. Retraction is essential for a student model because students are constantly updating their beliefs. Moreover, the tutoring system might misunderstand a student. Misunderstandings should be corrected after the system obtains more information about the student. Existing student models can be generally classified into two groups: bug-rule models and overlay models [Self, 1988]. A bug-rule model is usually a short-term model which results from analysis of the student's responses in a single interaction. While most overlay models are intended to be long-term, they are usually incremental. Retraction in overlay models is difficult and not well studied [Huang, 1990a]. One of the two main goals of this paper is to characterize non-monotonic revision in a student model.

The other main goal of the paper is to describe a representation for default knowledge in student models (and in user models in general). Information about a user that an intelligent system can obtain from observations is usually far from complete. One possible way to handle incomplete knowledge in user models is to use default knowledge. Several systems have been developed using this approach. Among them, Nested Theorist (NT) provides a formal representation for default knowledge in a nested belief system [van Arragon, 1989]. Since a user model is actually a set of a user's beliefs nested in the system's beliefs, NT provides a tool to formalize user modelling systems. Another system, GRUNDY, develops a representation called *stereotypes* for knowledge packages [Rich, 1979]. In GRUNDY, different sets of stereotypes are activated for different users. Information in the active stereotypes is assumed true even if it is not directly observed. Thus, it provides a default user model. GUMS1 goes further in the direction of using stereotypes in user modelling, allowing revision of a user model [Finin and Drager, 1986]. In GUMS1 user knowledge is organized in a stereotype tree in which each node represents a class of users. The user model is revised when the application system observes new facts that conflict with the active stereotype. This is done by changing the active stereotype to the parent of the currently activated one. This treatment is often not appropriate, since after several interactions, even the root may conflict with the observed facts. Then the active stereotype becomes an empty set which is no longer useful. Also, a user model usually doesn't fit a single knowledge package in the knowledge base, but a combination of several knowledge packages. An alternative is that each combination is also stored as a package, but then the user model may contain thousands of packages. In the *Student Model Maintenance System (SMMS)* presented in this paper, a knowledge partitioning hierarchy, instead of a class hierarchy, called the *Default Package Network (DPN)*, is used to represent default knowledge of a student model.

The SMMS is built as a component of an overall student modelling system whose architecture is shown in Figure 1. As is mentioned above, the student model contains two kinds of knowledge about a student: *confirmed knowledge* and *default knowledge.* Confirmed knowledge is the tutor's beliefs about the student's knowledge obtained from analyses of the student's responses during interactions and inferences on these analysis results. These analyses and inferences are made by a *student knowledge analysing system*[1] in the student modelling system. Default knowledge is information about the student assumed by the system when some necessary evidence is absent. The Default Package Network (DPN) contains stereotypical information about knowledge of different types of students. Note that the SMMS is general and not restricted to work with any particular student knowledge analysing system or in any particular domain, although in the discussion of this paper it is assumed to work with the one in the SCENT-3 programming advisor [McCalla, *et al.*, 1989].

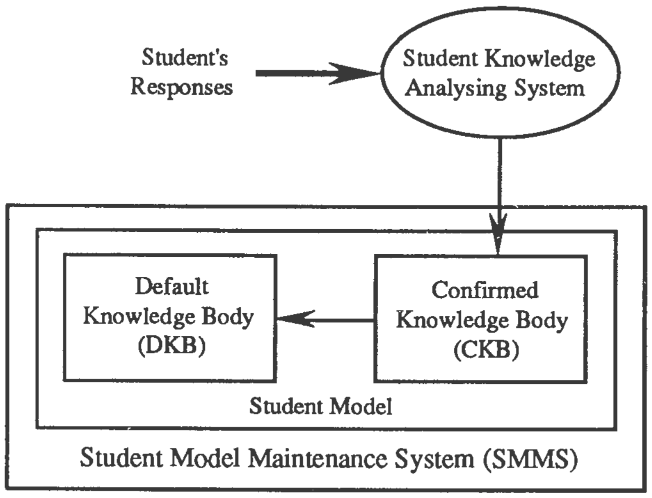

Student's Responses
Student Knowledge Analysing System
Default Knowledge Body (DKB)
Confirmed Knowledge Body (CKB)
Student Model
Student Model Maintenance System (SMMS)

Figure 1. Architecture of the Student Modelling System

The SMMS deals with two types of revision in a student model: evolutionary revision and revolutionary revision. Evolutionary revision is made to the *Confirmed Knowledge Body (CKB)* of the student model. An evolutionary revision occurs when the student knowledge analysing system obtains new information about the student's knowledge. In particular, if the new information conflicts with the old beliefs in the CKB, some beliefs must be removed in order to maintain consistency of the student model. This type of revision is accomplished by an *Evolutionary Belief Revision System (EBRS)* which combines techniques of truth maintenance systems [Doyle, 1979] [de Kleer, 1986] with diagnostic systems [Reiter, 1987] [de Kleer and Williams, 1987]. The EBRS is briefly described in Section 2. Its full details can be found in [Huang, *et al.*, 1989].

1 Student knowledge analysing systems are usually referred to as diagnostic systems in ITS. We use a new name here because these systems usually take more responsibility than and use different techniques from normal diagnostic systems in the artificial intelligence area such as those described in [Reiter, 1987] and [de Kleer and Williams, 1987]. The architecture of student knowledge analysing systems is another fundamental research topic in the ITS area (see [Clancey, 1986] and [Wenger, 1987]) which is not in the scope of the research in this paper.

Revolutionary revision occurs in the *Default Knowledge Body (DKB)* of the student model. It is triggered by an evolutionary revision in the CKB when it becomes obvious that a radical change in beliefs about the student's knowledge must be made. The system starts with an assumption about the student's knowledge status at each knowledge package (node) in the DPN. When the CKB is revised, the student model may no longer be consistent with some assumptions. These assumptions must be revised, which results in a new DKB. The DPN and revolutionary revision is discussed in Section 3. Section 4 of the paper presents a working example of the SMMS to show how these two types of revision are carried out. Finally, Section 5 summarizes the research results and proposes future research directions.

2. Revising Confirmed Student Knowledge: an Evolutionary Process

A student model is not a fixed knowledge body. A student constantly acquires new knowledge during learning. Also, the tutoring system knows more and more about the student as it interacts with the student. The student's new knowledge may conflict with old beliefs. The system may also obtain new information about the student which conflicts with its student model. Thus, re-establishing consistency of confirmed knowledge in the student model is necessary each time after new information is obtained. At first this might appear to require a truth maintenance system (TMS) such as developed in [Doyle, 1979] [de Kleer, 1986] and others. However, TMS's are designed to maintain consistency of *working hypotheses*. Confirmed knowledge of a long-term student model is a set of beliefs that have been accepted by the tutoring system. A working hypothesis and an *accepted belief* differ in that the former will be abandoned after the current problem is solved unless it is proved to be true and thus is accepted as a belief by the system, while the latter will not be abandoned until it is shown to be false or obsolete in the future. To revise the accepted beliefs of a system, simply maintaining consistency is not sufficient. What is also needed is that at any time after a change in beliefs, the set of new beliefs does not radically differ from the set of old beliefs, according to the *principle of conservatism* of accepted beliefs [Harman, 1986]. A TMS is not suitable here since it provides very little information about how to minimally modify the old beliefs to accommodate the new beliefs.

The SMMS uses the Evolutionary Belief Revision System (EBRS) described in [Huang, *et al.*, 1989] for revision of the confirmed knowledge body (CKB) of the student model. The EBRS combines TMS techniques and diagnostic techniques. It modifies the ATMS [de Kleer, 86] by identifying one of the consistent belief sets as the system's *belief set*. A diagnostic system is used in the EBRS to identify a set of minimal changes that remove all discovered conflicts from the belief set.

The EBRS records the system's confirmed beliefs about the student's knowledge, as well as data dependencies among the confirmed beliefs drawn from inferences. (In this section, we simply call a confirmed belief a *belief*, if no confusion is caused.) There are two kinds of beliefs: *premises* and *inferred* *beliefs*. Premises are beliefs that are considered true by the system without depending on other beliefs, but they are not immutable, that is, they subject to remove if they subsequently encounter conflicts. A premises is a belief directly extracted from the student's response or a rule employed to infer other beliefs. An inferred belief is a belief thus inferred.

Similar to the ATMS, a belief and its data dependencies in the EBRS are recorded in a data structure called an *EBRS node* which is of the form:

[datum, label, justifications].

The datum is a propositional sentence that represents the belief. A *justification* is a list of EBRS nodes that are used in an inference in which the belief is generated. The *label* is the set of minimal consistent *environments* in which the belief is eventually grounded, by tracing back via the justification links. An environment is a set of premises. In particular, a premise has itself as a justification. Its label contains a singleton which contains only itself. For example, assume that the student knowledge analysing system first obtains a belief, "knows(car/cdr-recursion)", from analysis of a student's response. Next it applies a rule, "knows(car/cdr-recursion) ---> knows(cdr-recursion)", to conclude another belief about the student, "knows(cdr-recursion)". We may use the logical symbols A and B to represent "knows(car/cdr-recursion)" and "knows(cdr-recursion)", respectively. Then the three sentences can be represented by the following EBRS nodes:

1. [A, {{1}}, {(1)}] 2. [A ---> B, {{2}}, {(2)}]
3. [B, {{1, 2}}, {(1, 2)}].

The EBRS keeps track of the system's belief set by identifying the set of premises from which the current belief set is inferred. This set of premises is called the *current environment*. A subset of the current environment is called an *active environment*. A propositional sentence in the knowledge base is currently believed by the system (thus is in the belief set) if and only if the label of its EBRS node contains an active environment.

An analysis of a student's response or an inference on the existing beliefs is called an *event*. The EBRS obtains a set of beliefs from the student knowledge analysing system after each event. A single event may generate many new beliefs. The EBRS puts these new beliefs into its knowledge base. Note that a new belief may not be *brand-new*, that is, it may have been generated in some previous events. The EBRS creates a new node for a new belief only if it is brand-new, but it always adds a new justification to a node that represents a new belief, unless the justification already exists. After a justification is added, label updating described in [de Kleer, 1986] propagates the effects of the new information, generating an updated label for each EBRS node in the knowledge base.

A new node may also be created when a contradiction between the new beliefs obtained in the event and the old beliefs in the knowledge base is detected. A node used to record a contradiction is called a *contradiction node*. In a contradiction node, the datum entry is "false" (noted by $\perp$). Environments in the label of a contradiction node are thus inconsistent environments. To remove a contradiction, simply deleting a justificant (a node in a justification) of the contradiction node is not sufficient, since the justificant may be supported by other nodes. Eventually, at least one premise in each active environment in the label of the contradiction node must be removed from the current environment, which causes the premise and some other beliefs directly or indirectly supported by the premise to be retracted from the belief set. If a contradiction node holds in several active environments or if many contradictions are detected after an event, the system may need to remove several premises from the current environment. These active environments may not be disjoint. Thus, retracting a premise may deactivate several inconsistent environments. According to the principle of conservatism, the set of retracted premises should be *minimal*, which means that if retraction of a set S can remove all contradictions, then no proper superset of S should be retracted instead.

Finding the collection of minimal sets of premises to retract in order to remove all contradictions is actually a diagnosis problem. In fact, the EBRS uses a modified version of Reiter's diagnostic algorithm [Reiter, 1987] to compute this collection. (The algorithm of the EBRS is described in [Huang, *et al.*, 1989].) Using the diagnostic algorithm, the EBRS quickly filters out unlikely alternatives of the set of retracted beliefs. However, identifying the most plausible candidate from these minimal sets usually requires domain knowledge, that is, educational knowledge in a tutoring system. The EBRS is a domain-independent knowledge base management system. It has to rely on the student knowledge analysing system, which has domain knowledge, to finally choose the set of retracted premises from the minimal sets.

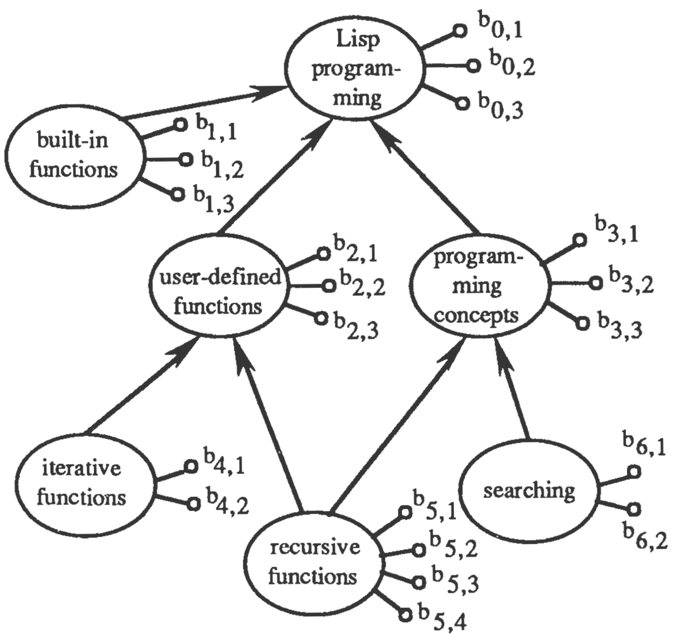
Note that the rules used by the student knowledge analysing system to infer student knowledge (e.g., node 2 above) belong to a tutor's educational knowledge. They are meta-knowledge which is distinct from the student model itself. Thus, the CKB of the student model is the belief set of the EBRS with the meta-rules removed.

3. Revising Default Student Knowledge: a Revolutionary Process

The tutoring system's default knowledge about students is represented in a *Default Package Network (DPN)*. A DPN is a directed acyclic graph. A node in a DPN represents a default knowledge package. Links are part-of relations, that is, the sub-graph of a node represents a part of the knowledge that the sub-graph of each of its parents represents. Figure 2 shows a segment of a DPN that represents default knowledge of Lisp programming.

Small circles attached to a node represent propositional sentences contained in the knowledge package that the node represents. Such a propositional sentence is called a *d-sentence*. In Figure 2, for example, the d-sentence "knows(S-expression-evaluation)" is labeled "S-expr evaluation" and denoted by $b_{0,1}$ (for it is the first d-sentence of V_0). A d-sentence is attached to the node representing the most general knowledge category relevant to it in order to minimize duplicate information. Thus, the d-sentence "knows(function-body)" (i.e, $b_{2,3}$) is attached to the "user-defined functions" node but not the "recursive functions" node. If a d-sentence is currently in the default knowledge body (DKB) of the student model, then it is a *d-belief*.

Corresponding to an estimate of the tutoring system about a student's knowledge status with respect to a package, a node in a DPN can be assigned a *node value* (also called *value* below if no confusion could be caused) in a designated value range (e.g., <NV, AV, EX> for "novice", "average" and "expert"). The value determines the d-belief set of the node. For example, if we assign the value EX to the node V_3, then all three d-sentences of it might be in the student model by default. If we assign AV to V_3, however, then only $b_{3,2}$ and $b_{3,3}$ might be d-beliefs. Thus, the DKB is determined by the current value assignment of the nodes of the DPN. To account for the case that the system has no idea about or no interest to the student's knowledge status in some package, a distinguished value "unknown" (denoted by UN) is defined. If a node has an "unknown" value, then no d-sentence of it would be a d-belief.



$b_{0,1}$: S-expr evaluation	$b_{0,2}$: calling functions
$b_{0,3}$: nested functions	$b_{1,1}$: car
$b_{1,2}$: cdr	$b_{1,3}$: cons
$b_{2,1}$: function name	$b_{2,2}$: parameters
$b_{2,3}$: function body	$b_{3,1}$: algorithms
$b_{3,2}$: data structure	$b_{3,3}$: efficiency
$b_{4,1}$: loops	$b_{4,2}$: local variables
$b_{5,1}$: recursive case	$b_{5,2}$: base case
$b_{5,3}$: reduction	$b_{5,4}$: action
$b_{6,1}$: search key	$b_{6,2}$: exploration

Figure 2. A Sample DPN for a Fragment of Lisp Programming Knowledge

An estimate (node value) for a student at a specific knowledge level may be made according to an estimate at a more general level. Thus, the value of a node may determine the value of a child of it by default. For example, if the student has an EX value in V_3, it might be reasonable to assign AV to V_5 and EX to V_6 (its two children) by default. On the other hand, the revised student model after an evolutionary revision (recall the discussions in Section 2) may not be consistent with the tutor's estimate of the student's knowledge status in a package. In this case, the tutor must change the estimate. Of course, the tutor might not have to change the estimate if the evolutionary revision does not cross the thresholds of the estimate. Thus, there is a set of ***constraints*** (i.e., thresholds) associated with each value of a node in the DPN. The value may be assigned to the node only if the student model satisfies this set of constraints. If the student model satisfies the constraint sets of several values of a node, then the value closest to the previous one should be chosen (which also implies that the value is only changed when necessary). The value of a node is not only constrained by the student model, but also by the values of its children. For example, we would not estimate a student to be an expert Lisp programmer if we believe that she/he has little knowledge of Lisp built-in functions. Of course, a value of a node need not have a constraint on the d-belief set and on every child of the node. In the case that the constraint is absent, the d-belief set can be any subset of the d-sentences of the node, while a child can have any value.

A list of a possible set of defaults and a set of constraints corresponding to each value of each node of the DPN in Figure 2 (except those for V_1 and V_6 which are not relevant in the following discussion) is shown in Table 1. In the table, we use the notions $\leq$, $=$ and $\geq$ to express constraints among the nodes. This is based on a designated total order among these values: NV < AV < EX. Also, SM, B_i and $b_{i,j}$ denotes the student model, the d-belief set of V_i and the jth d-sentence of V_i, respectively. Although constructing the set of defaults and constraints for a DPN is a knowledge engineering problem which depends on the domain of the application, some general principles could be useful. First, if $S \subseteq SM$ is a constraint and B the d-belief set for the same value of a node, then S should be a subset of B. Intuitively, one's defaults should not violate the constraints of one's estimate. Second, if x and y are two node values and $x < y$, then in each node the constraints of x should be strictly weaker (i.e., require less knowledge in the student model) than the constraints of y. That is, the requirement to be a less advanced student should be weaker than the requirement to be a more advanced student.

Changing the value of a node may cause the value of a child to be changed if the child's current value is "unknown" or was determined by the previous value of the node. For example, if V_4's AV value was determined by V_2's previous value AV and now V_2's value is changed to EX, then V_4's value would be changed to EX because of the default assignment of V_2's new EX value. Thus, value assignment in a DPN may propagate downwards. This process is called *default propagation*. On the other hand, if a node is forced to change its value since a constraint of its current value is violated (by the revised student model generated in an evolutionary revision or by a new value of its children), its new value may violate the value of its parents. Thus, value change may also propagate upwards. This process is called *constraint satisfaction*. For example, assume V_0 = EX, V_3 = AV, V_6 = AV. Now V_6 is forced to change its value to NV. This violates a constraint of V_3's AV value. Thus, V_3 is changed to NV as well, which in turn forces V_0 to change to AV. Note that when a value resulting from constraint satisfaction conflicts with a value resulted from default propagation, the result of constraint satisfaction has a higher priority since it comes from a more concrete information source (confirmed knowledge about the student).

Therefore, an evolutionary revision of a student model that occurs in its CKB may force value changes in some nodes of the DPN, causing changes of the default sets in the corresponding packages. This is a *local revolutionary revision* of the student model. In addition, value changes in some nodes may trigger the bottom-up constraint satisfaction process, followed by the top-down default propagation process, changing values of many nodes. Then, the DKB (and thus also the student model) undergoes a *global revolutionary revision*. The ratio of revolutionary revisions to evolutionary revisions depends on the tolerance of the constraint sets designed for the DPN.

Even if the constraint sets are designed carefully, there may be cases in which the student model doesn't satisfy the constraint set of any value of a node. This usually happens when a node, say V_i, is forced to change value during constraint satisfaction, while the new value to be assigned also has some constraint not satisfied (called a *second violation*). If the second violation comes from a child whose current value was determined by the previous value of V_i, then the second violation would be removed in the next default propagation, and thus ignored, so the new value is still assigned. Otherwise, V_i is assigned the "unknown" value. Here the "unknown" value means "unclassifiable", which may be slightly different from its original intuition "having no idea", but the same semantics applies. An important property of the "unknown" value is that it is a "wild card" value which can satisfy any constraints and that itself has no constraint. Thus, the value of a node is not changed when one of its parents or its children becomes "unknown". This has the advantage that failure of the system can be restricted to the local level, similar to what has been achieved in using a granularity hierarchy for recognition [Greer and McCalla, 1989].

Another use of the "unknown" value is to avoid circularity. During default propagation, a parent, V_a, of an "unknown" node, V_c, may propagate a value to V_c, while this new value of V_c may violate a constraint of another parent of V_c. This may generate a revision circle (see [Huang, 1990b] for details). To avoid this revision circle, we block the default propagation at V_c by letting it remain "unknown".

V_0:		
EX:	defaults:	$B_0=\{b_{0,1}, b_{0,2}, b_{0,3}\}$, V_1=AV, V_2=EX, V_3=EX;
	constraints:	$(\{b_{0,1}, b_{0,2}\} \subseteq SM) \vee (\{b_{0,1}, b_{0,3}\} \subseteq SM)$, $V_1 \geq AV$, V_2=EX, $V_3 \geq AV$;
AV:	defaults:	$B_0=\{b_{0,1}, b_{0,2}\}$, V_1=AV, V_2=EX, V_3=AV;
	constraints:	$\{b_{0,1}\} \subseteq SM$, $V_2 \geq AV$, $V_3 \leq AV$;
NV:	defaults:	$B_0=\{b_{0,1}\}$, V_1=UN, V_2=NV, V_3=AV;
	constraints:	$\{b_{0,3}\} \not\subseteq SM$, $V_2 \leq AV$, V_3=NV;
V_2:		
EX:	defaults:	$B_2=\{b_{2,1}, b_{2,2}\}$, V_4=EX, V_5=EX;
	constraints:	$\{b_{2,1}, b_{2,2}\} \subseteq SM$, $V_4 \geq AV$, $V_5 \geq AV$;
AV:	defaults:	$B_2=\{b_{2,1}, b_{2,2}\}$, V_4=AV, V_5=AV;
	constraints:	$(\{b_{2,1}\} \subseteq SM) \wedge (\{b_{2,3}\} \not\subseteq SM$, $V_4 \leq AV$, $V_5 \leq AV$;
NV:	defaults:	$B_2=\{\}$, V_4=NV, V_5=NV;
	constraints:	$(\{b_{2,2}\} \not\subseteq SM) \wedge (\{b_{2,3}\} \not\subseteq SM)$, V_4=NV, $V_5 \leq AV$;
V_3:		
EX:	defaults:	$B_3=\{b_{3,1}, b_{3,2}, b_{3,3}\}$, V_5=EX, V_6=EX;
	constraints:	$\{b_{3,2}, b_{3,3}\} \subseteq SM$, $V_5 \geq AV$, V_6=EX;
AV:	defaults:	$B_3=\{b_{3,2}, b_{3,3}\}$, V_5=AV, V_6=EX;
	constraints:	$(\{b_{3,1}\} \not\subseteq SM) \wedge (\{b_{3,2}\} \subseteq SM)$, $V_6 \geq AV$;
NV:	defaults:	$B_3=\{ \}$, V_5=NV, V_6=AV;
	constraints:	$(\{b_{3,1}\} \not\subseteq SM) \wedge (\{b_{3,2}\} \not\subseteq SM)$, $V_5 \leq AV$, $V_6 \leq AV$;
V_4:		
EX:	defaults:	$B_4=\{b_{4,1}, b_{4,2}\}$;
	constraints:	$\{b_{4,1}\} \subseteq SM$;
AV:	defaults:	$B_4=\{b_{4,1}\}$;
	constraints:	$(\{b_{4,1}\} \subseteq SM) \wedge (\{b_{4,2}\} \not\subseteq SM)$;
NV:	defaults:	$B_4=\{\}$;
	constraints:	$(\{b_{4,1}\} \not\subseteq SM) \wedge (\{b_{4,2}\} \not\subseteq SM)$;
V_5:		
EX:	defaults:	$B_5=\{b_{5,1}, b_{5,2}, b_{5,4}\}$;
	constraints:	$\{b_{5,2}, b_{5,4}\} \subseteq SM$;
AV:	defaults:	$B_5=\{b_{5,1}, b_{5,2}\}$;
	constraints:	$(\{b_{5,2}\} \subseteq SM) \wedge (\{b_{5,3}\} \not\subseteq SM)$;
NV:	defaults:	$B_5=\{b_{5,2}\}$;
	constraints:	$(\{b_{5,1}\} \not\subseteq SM) \wedge (\{b_{5,3}\} \not\subseteq SM)$.

Table 1. Defaults and Constraints of the DPN in Figure 2

Based on the above discussion, now we can summarize the algorithm of the student model revision process described in last and this sections:

<u>Algorithm SMMS:</u>

```
-- Compute the updated CKB (using the EBRS)
-- Compute the student model (from the updated CKB and
   the current DKB)
-- Satisfy the constraints in the DPN (bottom-up)
-- Propagate value changes along with the DPN according to
   the defaults (top-down)
-- If any change in the DPN is made, then
      * Compute the updated DKB (according to the updated
        DPN)
      * Compute the student model again (from the updated
        CKB and the updated DKB)
End.    {of the SMMS algorithm}
```

Each step of the algorithm is a procedure. The procedure for updating the CKB has been presented and discussed in detail in [Huang, *et al.*, 1989]. The DKB is the union of the d-belief sets of the nodes in the DPN. The student model is the union of the CKB and the DKB with the removal of every default belief in the DKB which directly contradicts a confirmed belief in the CKB (i.e., if p is in the DKB while $\neg p$ is in the CKB, then p is removed). Constraint satisfaction works upwards while default propagation goes downwards. In fact, the nodes in the DPN are numbered in a total order. Both procedures work according to this order. Constraint satisfaction starts at the node of largest number (a leaf) and ends at the node of smallest number (the root), while default propagation goes in the reverse direction.

Like other truth maintenance systems and diagnosis systems, the EBRS called at the first step requires exponential time. The remainder of the algorithm, however, requires only O(N log N) time for most student modelling systems and $O(M^2 + L \log L)$ for any kind of student/user modelling systems, where L is the size of the CKB, M the size of the DPN, and N = L + M. Details of the algorithm and a complexity analysis can be found in [Huang, 1990b].

4. An Example

This section integrates all that has been discussed in the paper in an example to show how the SMMS carries out an evolutionary revision in response to new information, how the evolutionary revision triggers a revolutionary revision, and how the revolutionary revision is realized in a DPN. The DPN used in the example is the one displayed in Figure 2. Its defaults and constraints are defined in Table 1. Now first assume at time t_0, before the revision happens, the knowledge base in the EBRS is as follows:

1. $[b_{0,1}, \{\{1\}\}, \{(1)\}]$
2. $[b_{2,1}, \{\{2\}\}, \{(2)\}]$
3. $[\neg b_{3,2}, \{\{3\}\}, \{(3)\}]$
4. $[b_{5,2}, \{\{4\}\}, \{(4)\}]$
5. $[b_{0,1} \dashrightarrow b_{3,1}, \{\{5\}\}, \{(5)\}]$
6. $[b_{3,1}, \{\{1, 5\}\}, \{(1, 5)\}]$;

and the current environment is {1, 2, 3, 4, 5}. Thus, all sentences in the knowledge base are believed (in the belief set). Since the CKB equals the belief set with the meta-rules removed (i.e. metarule: "$b_{0,1} \dashrightarrow b_{3,1}$"), we have

$$CKB(t_0) = \{b_{0,1}, b_{2,1}, b_{3,1}, \neg b_{3,2}, b_{5,2}\}.$$

We also assume that the value assignment to the DPN at time t_0 is:

$$V_0 = AV, \quad V_1 = UN, \quad V_2 = AV, \quad V_3 = NV,$$
$$V_4 = UN, \quad V_5 = AV, \quad V_6 = UN.$$

The DKB is the union of the d-belief sets of the nodes determined by the defaults of this value assignment. Thus, we have

$$DKB(t_0) = \{b_{0,1}, b_{0,2}, b_{2,1}, b_{2,2}, b_{5,1}, b_{5,2}\}.$$

The student model is the union of the CKB and the DKB with removal of each d-belief of the DKB that directly contradicts a confirmed belief in the CKB. Therefore,

$$SM(t_0) = \{b_{0,1}, b_{0,2}, b_{2,1}, b_{2,2}, b_{3,1}, \neg b_{3,2}, b_{5,1}, b_{5,2}\}.$$

At this moment, two new confirmed beliefs about the student's knowledge, $b_{5,3}$ and $b_{5,4}$, are obtained. Among the new beliefs, $b_{5,4}$ is obtained by applying a meta-rule, "$b_{3,1}, b_{5,3} \dashrightarrow b_{5,4}$". Thus, three new EBRS nodes are created:

7. $[b_{5,3}, \{\{7\}\}, \{(7)\}]$
8. $[b_{3,1}, b_{5,3} \dashrightarrow b_{5,4}, \{\{8\}\}, \{(8)\}]$
9. $[b_{5,4}, \{\{1, 5, 7, 8\}\}, \{(6, 7, 8)\}]$.

This triggers a student model revision process. Assume that the system immediately detects a contradiction between the new belief $b_{5,4}$ and the old belief $\neg b_{3,2}$. (Note that a contradiction is not necessarily a logical contradiction. It may be derived from any set of sentences that the system believes to be conflicting. The SMMS does not force the semantics of contradictions. The application system is free to define this.) Thus, a contradiction node is created:

cont-1. $[\perp, \{\{1, 3, 5, 7, 8\}\}, \{(3, 9)\}]$.

To remove the contradiction, at least one of the elements of the premise set {1, 3, 5, 7, 8} must be retracted. Assume that the node 3 (i.e., $\neg b_{3,2}$) is retracted (see [10] for how to choose the premises to be retracted). Then the new belief set in the EBRS is {1, 2, 4, 5, 6, 7, 8, 9}. By removing the meta-rules recorded in the nodes 5 and 8, now the CKB is

$$CKB(t_1) = \{b_{0,1}, b_{2,1}, b_{3,1}, b_{5,2}, b_{5,3}, b_{5,4}\};$$

and the student model from $CKB(t_1)$ and $DKB(t_1)$ (which is the same as $DKB(t_0)$) is

$$SM(t_1) = \{b_{0,1}, b_{0,2}, b_{2,1}, b_{2,2}, b_{3,1}, b_{5,1}, b_{5,2}, b_{5,3}, b_{5,4}\}.$$

This shows an evolutionary revision of the student model.

Next, revision of the DKB starts. By checking Table 1, one can find that a constraint for assigning AV to V_5 is violated since SM contains $b_{5,3}$ now. Thus, V_5 is upgraded and assigned an EX value. The constraint satisfaction propagates from V_5 to V_2 and V_3. The value of V_2 changes from AV to EX as well, since a constraint for V_2 to keep its AV value, "$V_5 \leq AV$", is violated. For V_3, a constraint of its NV value, "$V_5 \leq AV$", is also violated. However, an AV value cannot be assigned to it (nor can an EX value, of course), since SM does not contain $b_{3,2}$, which also violates a constraint for V_3 to have an AV value. Thus, V_3 is assigned UN, which means that the system cannot classify the student's knowledge status in this package. Although two children of V_0 have their values changed, no constraint for its AV value is violated. Thus, V_0 keeps the value unchanged.

Then the SMMS runs its default propagation procedure. Since V_2's first child V_4 has an UN value at this time, it is assigned an EX value according to the default of V_2's EX value. Thus, at time t_2 when the revision is completed, the value assignment to the DPN is

$$V_0 = AV, \quad V_1 = UN, \quad V_2 = EX, \quad V_3 = UN,$$
$$V_4 = EX, \quad V_5 = EX, \quad V_6 = UN.$$

By taking the union of corresponding d-belief sets of the nodes,

$$DKB(t_2) = \{b_{0,1}, b_{0,2}, b_{2,1}, b_{2,2}, b_{4,1}, b_{4,2}, b_{5,1}, b_{5,2}, b_{5,4}\}.$$

Finally, since $CKB(t_2) = CKB(t_1)$, the updated student model is

$$SM(t_2) = \{b_{0,1}, b_{0,2}, b_{2,1}, b_{2,2}, b_{3,1}, b_{4,1}, b_{4,2}, b_{5,1}, b_{5,2}, b_{5,3}, b_{5,4}\}.$$

Thus, the default beliefs are automatically revised in correspondence with the revision in the confirmed beliefs. This results in a new student model which is drastically different from the old one (see the difference between $SM(t_2)$ and $SM(t_0)$). In other words, a revolutionary revision has occurred in the student model. Note that although in the example each confirmed belief in the EBRS corresponds to a d-sentence in the DPN, this is not necessarily always so. The EBRS can obtain beliefs from the student knowledge analysing system that are not stored in the DPN, which means that the tutoring system can know more about an individual student than what it predicts for normal students, by direct observations and use of its knowledge to make inferences.

5. Conclusions

We have dealt with the revision problem in a long-term student/user model. We have shown how to use techniques of truth maintenance and techniques of formal diagnosis to accomplish revision of the tutor's beliefs about the student's knowledge. The DPN has been developed to account for the tutor's incomplete knowledge about the student. Revision of the tutor's default beliefs can be naturally and efficiently accomplished in the DPN. We have also shown how the new information that triggers an evolutionary revision in the confirmed beliefs may also cause a revolutionary revision in the default beliefs, resulting in a desirable updated student model. The SMMS we have developed is domain-independent. In fact, the research results in this paper can apply to a variety of student models and other user models.

The next stage of our research will focus on revision in the student's beliefs nested in the tutor's beliefs (called the *student's beliefs* for short). A student's beliefs are usually not entirely consistent. It is desirable for a student model to capture the property of limited consistency in the student's beliefs. We expect that a student model maintenance system which fulfils the three types of belief revision (revision in the student's beliefs, revision in the tutor's confirmed beliefs and revision in the tutor's default beliefs) will be very useful for intelligent tutoring systems [Huang, 1990a].

Acknowledgements

We wish to thank the University of Saskatchewan and the Natural Sciences and Engineering Research Council of Canada for financial support.

References

[Clancey, 1986] Clancey, W. J., Qualitative student models, Traub, J. F. (Ed.), *Annual Review of Computer Science* **1** (1986) 381-450.

[de Kleer, 1986] de Kleer, J., An assumption-based TMS, *Artificial Intelligence* **28** (2) (1986) 127-162.

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

[Doyle, 1979] Doyle, J., A truth maintenance system, *Artificial Intelligence* **12** (3) (1979) 231-272.

[Finin and Drager, 1986] Finin, T. and Drager, D., GUMS1: a general user modelling system, *Proceedings CSCSI-86*, Montreal, Canada (May 1986) 24-30.

[Greer and McCalla, 1989] Greer, J. E. and McCalla, G. I., A computational framework for granularity and its application to educational diagnosis, *Proceedings IJCAI-89*, Detroit, Michigan (August 1989) 477-482.

[Harman, 1986] Harman, G., *Change in View: Principles of Reasoning* (MIT Press, Cambridge, MA, 1986).

[Huang, *et al.*, 1989] Huang, X., McCalla, G. I. and Greer, J. E., *Belief Revision: An Evolutionary Approach*,Research Report 89-2, ARIES Laboratory, Department of Computational Science, University of Saskatchewan, Canada, 1989, also submitted to *Artificial Intelligence.*

[Huang, 1990a] Huang, X., *Belief Revision in Student Modelling*, Ph.D. Thesis Proposal, Department of Computational Science, University of Saskatchewan, Canada, 1990, to appear.

[Huang, 1990b] Huang, X., *Student Model Revision: Evolution and Revolution*, Research Report 90-1, ARIES Laboratory, Department of Computational Science, University of Saskatchewan, Canada, 1990.

[McCalla, *et al.*, 1989] McCalla, G. I., Greer, J. E. and the SCENT Research Term, Intelligent advising in problem solving domains: the SCENT-3 architecture, in Frasson and Gauthier (Eds.), *Intelligent Tutoring Systems: At the*

Crossroads of Artificial Intelligence and Education (Ablex, Norwood, NJ, 1989) 140-161.

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

[Rich, 1979] Rich, E., User modelling via stereotypes, *Cognitive Science* 3 (1979) 329-354.

[Self, 1988] Self, J. A., By passing the intractable problem of student modelling, *Proceedings ITS-88,* Montreal, Canada (1988) 18-24.

[Sleeman and Brown, 1982] Sleeman, D. and Brown, J. S. (Eds.), *Intelligent Tutoring Systems* (Harcourt Brace Jovanovich, 1982).

[van Arragon, 1989] van Arragon, P., User modelling with limited reasoning using defaults, Manuscript, University of Waterloo, Canada, August, 1989.

[Wenger, 1987] Wenger, E., *Artificial Intelligence and Tutoring Systems*, (Morgan Kaufman, Los Altos, CA, 1987).

Tailoring Definitions Using a Multifaceted User Model

Margaret H. Sarner
Sandra Carberry
Department of Computer and Information Sciences
University of Delaware
Newark, Delaware 19716, USA

Abstract

This paper presents a computational strategy for reasoning on a multifaceted user model to generate definitions tailored to the user's needs in a task-oriented dialogue. The strategy takes into account the user's current focus of attention in his partially constructed plan, his domain knowledge, and his receptivity to different kinds of information. A system that uses this strategy will generate definitions that appear natural and that represent cooperative, intelligent behavior.

1 Introduction

Analysis of naturally occurring information-seeking dialogues indicates that the information-provider's responses are influenced by his perceptions about the information-seeker and the goals and plans motivating the information-seeker's queries. If an expert consultation system's responses are to be viewed as cooperative and natural, then it must exhibit the same kind of behavior. The system will be most effective if its responses contain exactly the information that will be most helpful to the user in the given situation.

This paper presents a computational strategy for reasoning on a multifaceted user model to generate definitions tailored to the user's needs in a task-oriented dialogue. The user model is a dynamically constructed representation of the user's domain knowledge, task-related goals and plans, and receptivity to different kinds of information. The system constructs a definition by weighting both the strategic predicates that might be used to construct a definition and the propositions that might fill them. These weights are used to construct a definition that includes the information deemed most useful, using information of lesser importance as necessary to adhere to common rhetorical practices. This strategy reflects our overall hypothesis that beliefs about the appropriate content of a definition should guide selection of a rhetorical strategy, instead of the choice of a rhetorical strategy determining content.

2 Definition Content

In task-oriented expert-consultation dialogues, an information-seeker interacts with an expert to construct a plan for accomplishing a task. In naturally occurring task-oriented dialogues the expert often produces definitions, either in response to requests by the information-seeker or spontaneously. The examples below, which are taken from transcripts of a radio financial advice program, illustrate requests for definitions by direct question and by an elliptical phrase which is interpreted as a request.

(1) U: "Can you tell me what the money market is?"

(2) E: "I'd like to see you put that into two different Southern utilities."
U: "Southern utilities?"

Our goal has been to design a definition module that can produce intelligent answers to such requests as part of an expert consultation system.

We use the word "definition" in a broad sense as the explanation of a term. There are many ways in which the content of a definition can vary. It may, for example, identify the entity being defined as a member of a superclass, list some of its attributes, give examples of it, or say how it works. Our analysis of naturally occurring dialogues has identified over a dozen types of verbal components (Figure 1) that typically appear in definitions. We will use the term *strategic predicate* to refer to a type of component. Each strategic predicate corresponds to the relationship of an aspect of the entity being defined to the entity itself. Although strategic predicates are conceptually similar to rhetorical predicates [Gri75; McK85], we prefer the term *strategic* to place the emphasis on giving information in a way that will be useful rather than on rhetorical style.

Our transcript analysis suggests that the content of definitions depends on several factors, including the entity being defined, the information-provider's beliefs about the person receiving the definition, and the situation in which the definition is given. Salient characteristics of the information-seeker include his general knowledge, expertise in the area, and personal prefer-

Strategic Predicate	Description
Identification	An entity's generic class.
Property	Properties of an entity.
Component	Separate parts or components of the entity.
Substance	What the entity is made of.
Procedure	Temporal sequence of steps for performing an action involving the entity.
Generation	Cause-effect trace involving the entity.
Prerequisite	Enabling condition for an entity or process involving it.
Effect	The effect of an entity or process involving it.
Necessity	Why an entity must exist in a process.
Equivalence	Another, perhaps more familiar, name for the entity.
Negative	Something that the entity is *NOT*.
Example	An illustration, instance, or example.
Contrast	A comparison with a weaker case.
Analogy	Something analogous to the entity.
Background	Historical background of the entity.

Figure 1: Components of Definitions

ences. Salient characteristics of the situation include the information-seeker's partially constructed plan for achieving his domain goals and his current focus of attention in the plan. Human experts appear to consider all of these factors when making a decision about what to say. The information-provider's beliefs about the information-seeker's knowledge, goals, plans, and attitudes enable him to avoid giving information that is already known, to use terms that are familiar to the information-seeker, to construct explanations that address the information-seeker's needs in the given situation, and to provide information in a form compatible with the information-seeker's style of learning. If a computer system is to produce cooperative, intelligent responses that address the user's perceived needs, it should similarly reason on a multifaceted model of the user.

3 Generating Tailored Definitions

In the course of ongoing task-oriented expert-consultation dialogues, many occasions arise in which the expert must provide a definition. Analysis of naturally occurring dialogue indicates that the definitions generated by human information-providers vary according to the situation in which the definition occurs. This appears to be the result of three factors:

1. In task-oriented dialogues, the information-provider knows something about what the information-seeker is trying to accomplish and will generate definitions that help the information-seeker achieve his goals.
2. Whereas static definitions or responses to one-shot requests for definitions must assume a generic model for the information-seeker, responses to definition requests during an ongoing dialogue can take into account acquired beliefs about the information-seeker's specific domain knowledge.
3. Whereas static definitions and responses to one-shot requests for definitions must be generated all at once, dialogue allows the information-provider to produce what he thinks will be an acceptable definition and analyze the information-seeker's response to determine whether to elaborate on the definition.

If an expert consultation system is to be viewed as *cooperative*, *intelligent*, and *natural*, it must take the above factors into account. Otherwise it will not appear to be directed toward the user's goals (uncooperative), will not appear to make use of what the user already knows (unintelligent), and will not appear to take advantage of the fact that the interaction is ongoing, as opposed to one-shot (unnatural).

This section presents a new strategy for generating tailored definitions in an expert consultation system. The strategy relies on a dynamically inferred model of the user's underlying task-related plan and focus of attention in that plan, along with information about the user's domain knowledge and receptivity to different kinds of information.

4 The Multifaceted Model

The knowledge base of our system contains a generalization hierarchy, a plan library, and a lexicon. The generalization hierarchy is built using a knowledge representation system based on KL-ONE [Bra79]. The plan library contains a set of domain goals and plans for accomplishing them. These plans are hierarchical since they can contain subgoals and subactions that also have associated plans in the plan library. Figure 7 illustrates a sample domain plan. The plan library is used by the TRACK plan inference system [Car87] to build a model of the user's underlying task-related plan incrementally from an ongoing dialogue; the resultant beliefs are represented in a tree structure called a *context model*. Each node in this tree represents a goal or action that the user has considered. Except for the root, each of these goals and actions appears in the domain plan for accomplishing its parent action in the context tree. The context model can be expanded to arbitrarily many levels of detail by repeatedly replacing non-primitive goals and actions with associated plans which themselves contain constituent goals and actions. Figure 5 illustrates a sample context model. While the knowledge base of our definition system is domain-specific, the reasoning strategies are independent of the domain.

Our multifaceted user model has three parts. The first component, a model of the user's domain knowledge, can be used to construct definitions that will be understood by the user. The second, a model of the user's task-related goals and plans, can be used to generate definitions that will be useful in the given situation and are relevant to the user's perspective on the domain. The third, a model of the user's preferences in explanation types, can be used to include the kind of information, such as examples or analogies, that the user assimilates most readily.

The first component of the user model is a copy of the knowledge base with each node and link in the generalization hierarchy and each component of plans in the plan library marked with a value on a scale from 0 to 1 indicating how certain the expert is that the user is familiar with the entity. We are assuming that this component of the user model is maintained by a variant of the user modeling system described in [Kas87], and that markings are altered as the dialogue progresses and the system's beliefs about the user's knowledge change.

The second part of the user model is a context tree representing the system's beliefs about the user's partially constructed task-related plan and focus of attention in that plan. The context tree and focus of attention are maintained by TRACK.

The third component is a model of the user's receptivity to the various strategic predicates. Some people respond well to examples, whereas others learn better from explanations of how things work. A human expert can observe these differences and make use of his observations in giving explanations; an expert consultation system should be able to act in the same manner. We are assuming that the user modeling system will be able to determine receptivity and will represent it as weightings associated with the strategic predicates. There are several clues to the user's receptivity to different predicates. For example, if the user responds favorably to a definition, the weights on the predicates used in that definition might be increased; conversely, if the user finds a definition unsatisfactory, weights on the predicates involved might be decreased.

5 Reasoning on the User Model to Generate Tailored Definitions

When a need for a definition of a term is recognized in the course of an expert-user dialogue, the definition module is activated. The strategy it uses for constructing a definition has four steps:

1. The predicates representing the kinds of information that might comprise a definition are weighted according to a model of the user's receptivity.
2. The knowledge base is searched for propositions that can be used to fill the predicates. Each proposition is evaluated according to the significance of its information at this point in the dialogue.
3. The propositions are divided into categories according to their estimated usefulness to the user, measured as a combination of predicate weight and proposition significance.
4. The definition is constructed to include the information deemed most important, using information of lesser importance as necessary to adhere to common rhetorical practices.

The first two steps will be described more fully in the rest of this section.

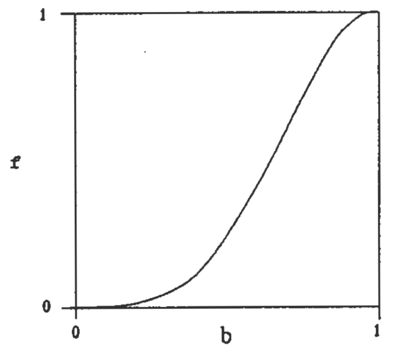
When a definition occasion arises, a local predicate receptivity model is created. The predicates are initially assigned weights that reflect the user's receptivity to different kinds of information as represented in the user model. These weights are adjusted by other factors, such as the type of question and the user's level of local domain knowledge. This latter weighting is suggested by Paris' findings that naive and expert users respond best to different kinds of explanations [Par88].

Semantics associated with each strategic predicate indicate where to look in the knowledge base for propositions that can be used to instantiate the predicate. Propositions that fill some predicates are found in the generalization hierarchy; for other predicates, candidate propositions come from the plan library.

At the same time as the candidate propositions are identified, a measure of significance is computed for each proposition. Significance is a function of two parameters: familiarity, which estimates how likely the user is to understand the information and is a function of the belief factors associated with entities in the proposition, and relevance, which estimates how useful the information is in the particular situation and is based on closeness to the current focus of attention in the context tree.

Familiarity for an entity is a function of the belief factor attached to its node in the user model. If the expert believes strongly that the user knows about the entity represented by the node, the belief factor is close to 1 and the familiarity component of the significance equation is high. Since the dialogue is ongoing and there will be an opportunity for the expert to make clarifications, nodes can be treated as if they are familiar even if the belief factors associated with them are only *close* to 1. As the expert's confidence diminishes, however, the value of the familiarity component decreases rapidly, since the motivation for considering the familiarity of the entity is to satisfy the criterion of making definitions in terms that are understandable to the user. This behavior is described by the curve shown in Figure 2. The formula $f = \frac{e^{6b(2-b)}-1}{e^6-1}$ where f is the familiarity rating and b is the belief factor, exhibits an appropriate amount of curvature to reflect the rapid drop-off in usefulness as the belief factor decreases. This equation has been formulated to capture the information in the curve so that the familiarity component for a given belief factor can be easily estimated.

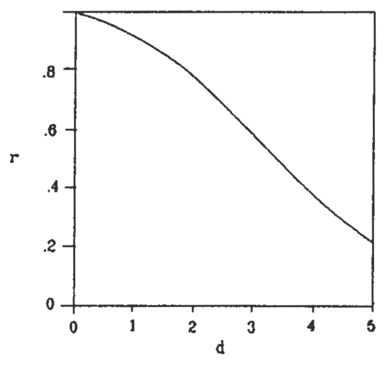
Relevance of an entity is a function of the relationship of the entity to the existing dialogue context. The context model is a hierarchical tree structure of goals and actions representing the system's beliefs about the user's task-related plan. One of the actions in the context model, along with its associated domain subplan, is marked as the aspect of the task on which the user's attention is currently focused. The components of this



b = belief factor
f = familiarity

Figure 2: Familiarity Curve

domain subplan, including the entities referenced in it, are of very high relevance. However, a focus shift occurs when attention is shifted either to more detailed actions that are part of its expanded subplan or back to higher-level actions whose performance includes the current focused action. Information in a subplan for an action that is one shift of focus away from the current focused action is still of high relevance. If the subplan is several focus shifts away, its relevance value and that of its parts is somewhat smaller, but as long as a plan has been activated, the information found in it is of some relevance. This situation, in which relevance remains high close to the focus of attention but drops off as the distance increases, is modeled by the curve shown in Figure 3. The equation $r = e^{-(\frac{d}{4})^2}$ where r is the relevance rating and d is the number of shifts from the current focus of attention, captures the desired features. This equation allows us to estimate relevance given the number of focus shifts between the current focus of attention and the appearance of the entity in an expansion of the context model. Some nodes in the generalization hierarchy may represent entities not mentioned explicitly in a plan. Relevance values for these nodes can be assigned using linear interpolation on the relevance values of other nodes on the isa chain.



d = focus shifts
r = relevance

Figure 3: Relevance Curve

Familiarity and relevance for a proposition are based on the familiarity and relevance values for information contained in the proposition. Both the metrics for estimating the familiarity and relevance of propositions and the methods for combining these metrics to compute the significance of a proposition depend on the strategic predicate being filled. Familiarity is weighted more heavily for predicates that describe the entity being defined in terms of something else, such as its parent in the generalization hierarchy, since the principal use of these components is to locate the entity with respect to something in the user's own knowledge base. Relevance is weighted more heavily if the predicate is used to tell something about the entity being defined, such as its properties, since aspects of the entity useful to the user will be those that are relevant to the plan that the user is constructing.

A closer look at the semantics for two of the strategic predicates will help illustrate how the selection and weighting of propositions is accomplished.

5.1 The Identification Predicate

The Identification predicate is used to identify an entity as a member of a particular class. For example, *"Baking soda is an antacid"* is an Identification definition.

The semantics for the Identification predicate indicate that propositions of this type should relate the entity being defined to one of its ancestors in the generalization hierarchy. A portion of the generalization hierarchy that includes baking soda is shown in Figure 4. The *isa* links between nodes are used to connect classes to superclasses. An *isa chain* is a chain of nodes connected by isa links, such as *Antacid*, *Medicinal Remedy*, *Substance*, *Thing*. A *relevant isa chain* is an isa chain that includes all relevant ancestor nodes – that is, nodes above the entity being defined that have at least some minimum relevance to the current focus of attention in the dialogue. A candidate Identification proposition is produced for every node in the relevant *isa* chain.

Note that a node can have more than one parent; for example, baking soda is a leavening agent, an antacid, and a deodorizer. Since it is quite possible that not all ancestor nodes will be relevant to the perspective from which the entity is being discussed, it is necessary to select the ones that are appropriate to use. If there is a branch point, then a decision must be made as to which ancestor belongs on the relevant isa chain. Our context model, representing the system's beliefs about the user's underlying task-related plan, captures how the user is viewing the domain. For example, if the user is constructing a domain plan for relieving indigestion, then baking soda should be viewed as an antacid, since an entity of type antacid can play a role in such a plan and baking soda can serve as that entity. On the other hand, if the user is constructing a domain plan for baking a cake, then baking soda should be viewed as a leavening agent. Therefore, the plan in which an entity plays a role determines the perspective from which the entity should

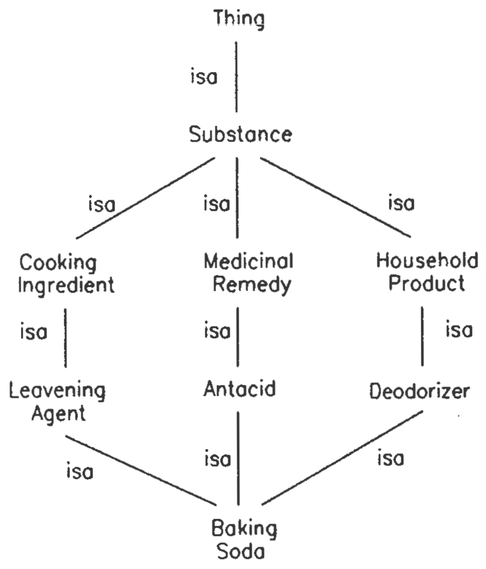


Figure 4: Portion of a Generalization Hierarchy

be viewed. We annotate the arguments of subactions in the body of plans in the plan library so that they contain type information specifying the relevant ancestors in the generalization hierarchy. This type information can be used to determine relevant isa chains for entities that appear in an expansion of the context model. Our selection of relevant isa chains is more limited in scope than McCoy's perspectives [McC88] which partition the knowledge base into overlapping segments to represent information salient to a particular viewpoint. Selecting a relevant isa chain for an entity only identifies the ancestor classes that are relevant to the user's perspective but does not identify salient properties of the entity. The latter is done by reasoning on expansions of the system's context model.

The familiarity and relevance values for an Identification proposition are the familiarity and relevance of the ancestor node selected from the generalization hierarchy. The significance calculation for Identification propositions is

$$significance = (2 \times familiarity + relevance) \div 3$$

Familiarity is weighted more heavily than relevance, since an identification is given in an attempt to tie the entity being defined to something the user already understands.

5.2 The Effect Predicate

An Effect definition describes the results of an action involving the entity being defined. In contrast with Generation, which provides a cause-effect chain of events leading to some state, the Effect strategic predicate describes a single effect or a conjunction of effects. *"Baking soda makes the cake rise"* defines baking soda in terms of an effect.

The semantics for the Effect predicate indicate that propositions are formed from effects of plans whose actions involve the entity. If actions involving the entity appear in several subplans relevant to the current dialogue context, the Effect predicate will be filled multiple times.

Sometimes it is more appropriate to give the definition in terms of indirect effects. Consider the example outlined below.

Plan-1:	Body-1:	Add baking soda
	Effect-1:	Release CO_2
Plan-2:	Body-2:	Release CO_2
	Effect-2:	Make cake rise

Two Effect definitions of baking soda could be derived from this set of plans. The definition based on the direct effect of adding baking soda (Plan1) would be *"Baking soda releases CO_2."* Alternatively, an indirect effect (the effect of releasing CO_2 shown in Plan2) would lead to the definition *"Baking soda makes the cake rise."*

Choices between direct and indirect effects are made according to a set of rules:

Rule 1. If all effects at one depth in a chain of effects have higher significance than effects at any other depth, choose those effects.

Rule 2. If all effects at one depth in a chain of effects are of greater or equal significance compared to effects at any other depth, with at least one effect having greater significance, choose those effects.

Rule 3. If there is a tie for greatest significance, select the effects from the plan that is closest to the plan containing the entity being defined, since these effects are more directly associated with the entity.

The purpose of these rules is to make choices between possible Effect propositions based on their appropriateness to the current situation. If multiple candidates remain after the rules are applied, the choice will be made later according to rhetorical considerations.

The familiarity of each entity involved in a proposition filling the Effect strategic predicate is based on its belief factor, as described earlier. Familiarity of the proposition is computed as the average of the familiarities of the entities comprising the proposition. Relevance is a function of how close the subplan containing the proposition is to the current focus of attention in the context model. Significance for Effect propositions is computed as

$$significance = (familiarity + 2 \times relevance) \div 3$$

Significance is based more heavily on relevance than on familiarity, since the appropriateness of the propositions filling the Effect predicate is tied to the user's plans.

6 Examples

The examples in this section will illustrate how definitions constructed by our system will vary depending on the system's beliefs about the user and the current situation as captured in its multifaceted user model.

Consider the following short dialogue segment between the system (S) and a naive traveler (U) who is unfamiliar with travelers checks:

U: "I am going to visit Paris. What must I have to pay for my hotel?"
S: "You will need to have a major credit card, travelers checks, or French currency."
U: "Travelers checks?"

Figure 5 illustrates part of the user's partially constructed plan inferred by the TRACK system. The focus of attention when the definition module is invoked is represented by the node preceded by an asterisk, with the most recently considered subgoal in the plan preceded by a bullet. The portion of the belief model that contains the concept of travelers checks is shown in Figure 6. The numbers indicate the belief factors attached to the nodes and links. For simplicity, we have limited the markings to 0 (definitely believed unfamiliar to the user) and 1 (definitely believed known by the user).

Instrument Payable To Order, *Money Instrument*, *Financial Thing*, and *Thing* form a relevant isa chain because they are ancestors in the generalization hierarchy of Travelers Check, which is part of the current focused plan in the context tree. Thus, the Identification predicate has several possible fillers. *Money Instrument* is familiar to the user and close to the concept of travelers checks which is part of the plan for the current focused action in the context model. Therefore, the significance value for the associated proposition, which is based on the familiarity and relevance of this node, is also high.

Another candidate proposition comes from the Prerequisite predicate. A domain plan for purchasing travelers checks (not shown), whose effect is to have travelers checks, has a precondition of being at a bank. Since this domain plan achieves the most recently considered subgoal (have travelers checks) in the user's current focused plan and since the system believes the concept of being at a bank is familiar to the user, the Prerequisite predicate filled with this proposition (and the name of the action, *purchase travelers checks*, associated with the domain plan) would also receive a high rating. These two predicates would produce the response *"Travelers checks are money instruments which you purchase at a bank."*

On the other hand, suppose that the dialogue segment were

U: "I am going to visit Paris. I am afraid of carrying a lot of cash."
S: "You can carry a major credit card or travelers checks."
U: "Travelers checks?"

As a result of the user's first utterance, the expert introduces plans that are relevant to visiting Paris and whose applicability conditions include wanting to avoid carrying a lot of cash. When the definition module is invoked, the focus of attention in the context model is on the action whose domain plan is shown in (Figure 7). The Effect predicate can be filled with the propositions Have-Convertible-Funds(_a:&TRAVELER, _amt:&AMOUNT) and Have-Safe-Funds(_a:&TRAVELER, _amt:&AMOUNT) along with the action, *carry travelers checks*, that generates these effects. Both of these propositions will be assigned high significance values since they consist of concepts that the system believes the user is familiar with and are part of the subplan in the context model on which the user's attention is currently focused. The two propositions filling the Effect predicate would be represented in the response *"Carrying travelers checks lets you have convertible funds in a safe form."*

In both of the above examples, the information that might be included in a response is evaluated according to its usefulness to the user in the given situation. Consequently, the definition that results will be tailored to the user's inferred plans and goals and the user's familiarity with domain concepts.

7 Categorizing Candidate Propositions

Once weights have been assigned to the candidate propositions, they are ranked according to weight and put into categories. There are four categories:

Must Say
Say if Convenient
Say if Needed for Coherence
Do Not Say

The higher wieght categories receive the higher-weighted propositions; the lower-weighted propositions go into the lower weight categories. Some categories may be empty.

When all category assignments have been made, the resulting four groups of propositions are passed to an answer generator which attempts to find a way to say all of the Must Say propositions and as many as possible of the Say if Convenient propositions, using Say if Needed for Coherence propositions whenever they help the construction of the response. We propose to do this task using rules of combination developed to produce an utterance that adheres to common rhetorical practices that people appear to follow. Since we are emphasizing saying the things that are most important to say and since it will happen that there are more important things to say about some subjects than about others, it will follow that some responses will be longer than others. This situation is consistent with definitions that occur in natural dialogues.

8 Related Work

Several researchers [Chi88; PM87] have concentrated on modeling the user's domain knowledge or expertise with the aim of giving the user information appropriate to his level of expertise or simply avoiding giving redundant information. Hovy [Hov87] used a model

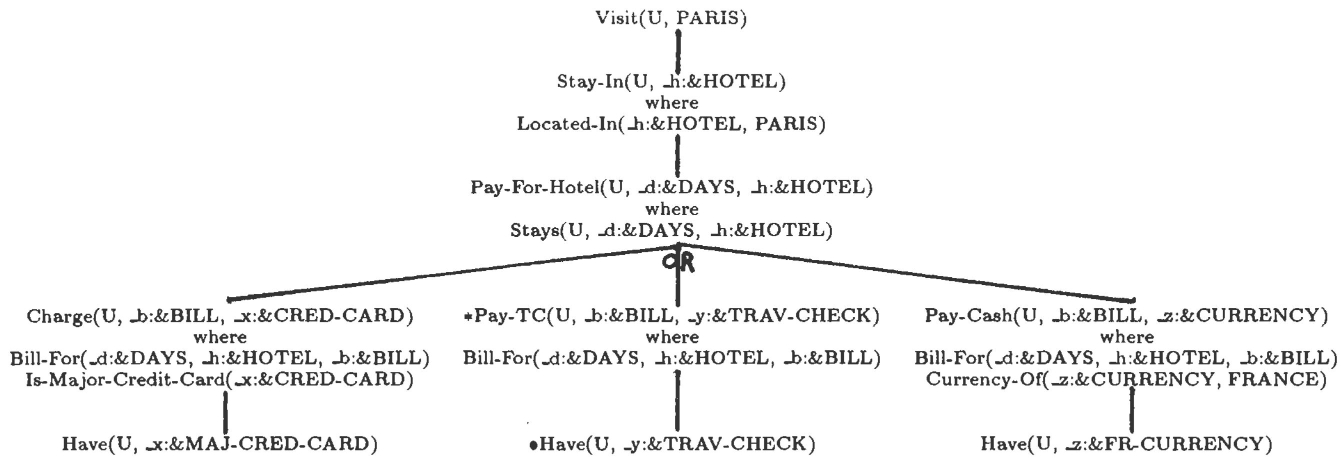

Visit(U, PARIS)
Stay-In(U, _h:&HOTEL)
where
Located-In(_h:&HOTEL, PARIS)
Pay-For-Hotel(U, _d:&DAYS, _h:&HOTEL)
where
Stays(U, _d:&DAYS, _h:&HOTEL)
OR
Charge(U, _b:&BILL, _x:&CRED-CARD)
where
Bill-For(_d:&DAYS, _h:&HOTEL, _b:&BILL)
Is-Major-Credit-Card(_x:&CRED-CARD)
*Pay-TC(U, _b:&BILL, _y:&TRAV-CHECK)
where
Bill-For(_d:&DAYS, _h:&HOTEL, _b:&BILL)
Pay-Cash(U, _b:&BILL, _z:&CURRENCY)
where
Bill-For(_d:&DAYS, _h:&HOTEL, _b:&BILL)
Currency-Of(_z:&CURRENCY, FRANCE)
Have(U, _x:&MAJ-CRED-CARD)
•Have(U, _y:&TRAV-CHECK)
Have(U, _z:&FR-CURRENCY)

Figure 5: The System's Context Model

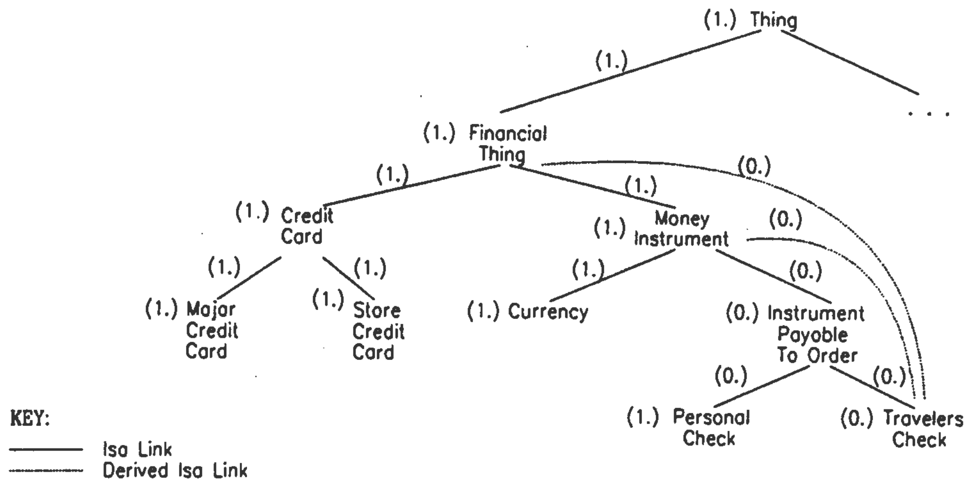


Figure 6: Portion of a Belief Model

of multiple characteristics of the user to select rhetorical strategies likely to achieve desired effects, and Bateman and Paris [BP89] developed a strategy for varying the phrasing of a proposition to make it appropriate for different types of users. Moore [Moo89] investigated a reactive model of explanations that enabled the system to reason about its own responses in addressing user misunderstandings.

In addition, van Beek and Cohen [vB87] explore appropriate responses for situations in which the user's plan would fail to achieve his goal or was not the most effective means for achieving it. McKeown [MWM85] used domain goals to index into a pre-built set of perspectives from which to answer *can* and *should* questions, and Cohen et al.[CJS*89] developed a strategy that took into account both a user's background knowledge and goals in generating answers to queries in an educational diagnosis system.

Although our research has a flavor similar to the above efforts, it differs from them in several ways. We are not concerned with generating answers to specific questions about the task at hand or the plan that the user is trying to construct. Instead, we are addressing the problem of generating definitions tailored to the user's needs in a task-oriented dialogue. Our strategy takes into account many different facets of the user, including his domain knowledge, level of expertise, task-related plans and goals, and receptivity to different kinds of information. It reasons directly on the system's beliefs about the user and the situation, without using pre-built perspectives, to propose and evaluate information that might be included in a response; an effective and coherent definition will then be constructed by taking into account both the significance of individual pieces of information and rhetorical considerations.

9 Conclusions

A subset of the predicates for the tailored definition system have been implemented. We are currently

```
Name: Carry-Convertible-Funds(_a:&TRAVELER, _amt:&AMOUNT)
Applicability Conditions:
    ¬Want[_a:&TRAVELER,
            Carry(_:&TRAVELER, _amt:&AMOUNT, _c:&CASH)
                          where
                 Is-Large(_amt:&AMOUNT)]
Preconditions: None
Body:
    Carry(_a:&TRAVELER, _amt:&AMOUNT, _t:&TRAV-CHECK)
Effects:
    Have-Convertible-Funds(_a:&TRAVELER, _amt:&AMOUNT)
    Have-Safe-Funds(_a:&TRAVELER, _amt:&AMOUNT)
```

Figure 7: A Domain Plan for Carrying Convertible Funds

working on implementing the remaining predicates, designing the answer generator that produces the final definition, and building a sufficiently large knowledge base to test the system. This definition module is being developed as part of the Delaware Intelligent Advisory Language System (DIALS).

The motivation for this work has been the hypothesis that responses should be tailored to the particular user and the current situation. We have presented a strategy for generating tailored definitions that takes into account the user's current focus of attention in his partially constructed plan, his domain knowledge, and his receptivity to different kinds of information. A system that uses this strategy and reasons on a multifaceted user model will generate definitions that appear natural and that represent cooperative, intelligent behavior.

References

[BP89] John Bateman and Cecile Paris. Phrasing a text in terms the user can understand. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1511–1517, Detroit, Michigan, 1989.

[Bra79] R. Brachman. On the epistemological status of semantic networks. In N. Findler, editor, *Associative Networks: Representation and Use of Knowledge by Computer*, Academic Press, N. Y., 1979.

[Car87] Sandra Carberry. Pragmatic modeling: toward a robust natural language interface. *Computational Intelligence*, 3:117–136, 1987.

[Chi88] David N. Chin. Knome: modeling what the user knows in uc. In Alfred Kobsa and Wolfgang Wahlster, editors, *User Models in Dialog Systems*, Springer Verlag, 1988.

[CJS*89] Robin Cohen, Marlene Jones, Amar Sanmugasunderam, Bruce Spencer, and Lisa Dent. Providing responses specific to a user's goals and background. *International Journal of Expert Systems*, 1989. To appear.

[Gri75] J. E. Grimes. *The Thread of Discourse*. Mouton, 1975.

[Hov87] Eduard H. Hovy. Generating natural language under pragmatic constraints. *Journal of Pragmatics*, 1987.

[Kas87] Robert Kass. *Implicit Acquisition of User Models in Cooperative Advisory Systems*. Technical Report MS-CIS-87-05, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, 1987.

[McC88] Kathleen F. McCoy. Reasoning on a highlighted user model to respond to misconceptions. *Computational Linguistics*, 52–63, 1988.

[McK85] Kathleen R. McKeown. *Text Generation*. Cambridge University Press, 1985.

[Moo89] Johanna Moore. Responding to huh?: answering vaguely articulated follow-up questions. In *Proceedings of the Conference on Human Factors and Computing Systems*, Austin, Texas, 1989.

[MWM85] K. McKeown, M. Wish, and K. Matthews. Tailor-
ing explanations for the user. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 794–798, Los Angeles, California, 1985.

[Par88] Cecile L. Paris. Tailoring object descriptions to a user's level of expertise. *Computational Linguistics*, 14(3):64–78, 1988.

[PM87] Cecile L. Paris and Kathleen R. McKeown. Discourse strategies for descriptions of complex physical objects. In G. Kempen, editor, *Proceedings of the Third International Workshop on Natural Language Generation*, 1987.

[vB87] Peter van Beek. A model for generating better explanations. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 215–220, Stanford, California, 1987.

Deriving Natural Language Presuppositions from Complex Conditionals*

Robert E. Mercer
Department of Computer Science
University of Western Ontario
London, Ontario, CANADA
N6A 5B7

Abstract

The application of Default Logic to the generation of natural language presuppositions for the particular class of natural language sentences called complex conditionals is demonstrated. Complex conditionals are conditionals which have embedded structure (for example, '*If A or B then C*' or '*If A then if B then C*'). A critical element in the generation of natural language presuppositions using Default Logic is the *clausal quantity implicature* ([Gazdar, 1979]). In order that this method can be applied to complex conditionals, the notion of clausal quantity implicature needs to be strengthened. The method to generate the natural language presuppositions previously described in [Mercer, 1987; Mercer, 1988b] remains unchanged.

1 Introduction

In this paper I describe the application of Default Logic to the generation of natural language presuppositions (henceforth presuppositions). In particular I focus on a problem, first described in [Soames, 1982], that concerns a class of sentences called complex conditionals. Complex conditionals are conditionals ('*If A then B*') which have embedded structure (for example, '*If A or B then C*' or '*If A then if B then C*'). Whereas previously I have concentrated on the motivation, success, and power of applying Default Logic to the generation of presuppositions [Mercer, 1987; Mercer, 1988b], what follows is an important modification of some inputs to this aforementioned theory.

Although the ideas first presented in [Mercer and Reiter, 1982], and later expanded in [Mercer, 1987; Mercer, 1988b] are basically sound, our understanding of subsidiary issues has needed refinement (for example, [Mercer, 1988a] improves our representation of sentential adverbs). A critical element in the generation of presuppositions using Default Logic is the *clausal quantity implicature* [Gazdar, 1979]. Here, the major contribution is that the notion of clausal quantity implicature is strengthened in order to deal with complex conditionals.

*This research was supported by NSERC grant 0036853.

Sections 2, 3, and 4 describe presuppositions, clausal quantity implicatures, and the problem of complex conditionals, respectively. Section 5 is a short description of the use of Default Logic to capture the class of linguistic inferences called presuppositions. Section 6 describes the generation of clausal quantity implicatures that are stronger than Gazdar's.

2 Presuppositions

In its linguistic sense, *presuppositions* are those inferences, generated from a number of linguistic situations, which pass a negation test, that is, being implied by a natural language sentence and the natural (or preferred) interpretation of its simple negation. Presuppositions are generated from lexical and syntactic contexts. Those contexts which pass the negation test can be termed *presuppositional environments*. Sentences (1)–(5) demonstrate some prototypical examples of presuppositions produced by the following presuppositional environments, respectively: noun phrases, possessives, factive verbs, certain aspectuals, and definitions of words. In each of these examples the truth of the affirmative a-sentence always implies the truth of the c-sentence, and the truth of the negative b-sentence normally implies the truth of the c-sentence.

(1) a. The present king of Buganda is bald.
b. The present king of Buganda is not bald.
c. There exists a present king of Buganda.

(2) a. Jack's children are bald.
b. Jack's children are not bald.
c. Jack has children.

(3) a. Mary is surprised that Fred left.
b. Mary is not surprised that Fred left.
c. Fred left.

(4) a. (At time t), John stopped beating the rug.
b. (At time t), John did not stop beating the rug.
c. (Prior to time t), John had been beating the rug.

(5) a. My cousin is a bachelor.
b. My cousin is not a bachelor.
c. My cousin is a male adult.

For many years the topic of debate has been what happens when presuppositional environments are found in compound[1] sentences. Since the main concern here is with complex conditionals, the discussion that follows concentrates only on compound sentences which are composed of sentences combined with logical functors.

Sentences combined with *or* and *if ... then*, sometimes display the presuppositions associated with the presuppositional environments found in both clauses, for example (6), and sometimes do not. For example, the presupposition associated with the presuppositional environment found in the consequent is not a presupposition of (7).

(6) Mary stopped beating the rug or John stopped beating the egg.

(7) If John was beating the egg then he has stopped (beating the egg).

The class of sentences on which this paper focusses is the complex conditional. (8) and (9) are examples from this class. (8) presupposes (among other things[2]) that '*my cousin is adult and male*' and that '*my teacher is adult and female*'. (9) presupposes that '*my cousin is adult*' and presupposes neither '*my cousin is male*' nor '*my cousin is female*'.

(8) If my cousin is a bachelor or my teacher is a spinster, someone at my party is unmarried.

(9) If my cousin is a bachelor or (my cousin is) a spinster, then my cousin will be the life of the party.

Most of the debate has centred around the appropriate system for producing the correct presuppositions for these sentences. [Mercer, 1987] contains a discussion of this debate and a system based on Default Logic which addresses the issue. For present purposes what follows focusses on two issues: the importance of *clausal quantity implicatures* in generating the correct presuppositions, and the role that these implicatures play in the Default Logic system itself.

3 Clausal Quantity Implicatures

Gazdar [1979] argues that if a speaker were to utter a compound sentence having a constituent which is not itself (or its negation) entailed or potentially presupposed,[3] then the speaker would be in breach of Grice's maxim of quantity if he knew that sentence to be true or false, but did not indicate to the listener that it was so, since the speaker could have been more informative by producing a compound sentence having the constituent concerned (or its negation) as an entailment or a presupposition. It follows that uttering such a compound sentence potentially implicates[4] that both the constituent sentence and its negation are compatible with what the speaker knows.

It follows from this argument (and the formal definition of *clausal quantity implicature* (see [Gazdar, 1979]:61) that the sentences '*A or B*' and '*If A then B*', where '*A*' and '*B*' are not compound, have the potential clausal quantity implicatures $\mathsf{P}_S A$, $\mathsf{P}_S \neg A$, $\mathsf{P}_S B$, and $\mathsf{P}_S \neg B$.

Clausal quantity implicatures are very important in both Gazdar's and the Default Logic theories of presupposition. For Gazdar they are used to cancel unwanted potential presuppositions. For the Default Logic theory they define the default theories in which the presuppositions are generated. That these clausal quantity implicatures, as described above, are not sufficient for complex conditionals has been previously noticed ([Soames, 1982] for Gazdar's system; [Mercer, 1987] for the Default Logic method). A discussion of this situation follows immediately and a more detailed account as it pertains to the Default Logic method is provided in section 5.2.

4 The Problem

The first indication of the shortcomings of the method proposed in [Gazdar, 1979], the inability of Gazdar's method to generate the appropriate presuppositions from conditionals containing sentential adverbs, appears in [Soames, 1979]. [Landman, 1981] suggests an ill-fated strengthening of the clausal quantity implicatures used in Gazdar's method (discussed briefly below) to overcome this problem.[5] As well as providing methodological arguments, [Soames, 1982] provides complex conditionals as the set of counterexamples to Landman's proposed modification. Soames' examples include (10). (8) and (9) display a similar structure. Complex conditionals are counterexamples to Gazdar's method, as well. Because the Default Logic method uses Gazdar's clausal quantity implicatures, the complex conditionals are counterexamples to this method, too.

(10) If the dress Mary bought is powder blue and the dress Susan bought is, too, then Mary will regret having bought a dress that is the same colour as Susan's.

Gazdar's method sometimes requires clausal quantity implicatures to cancel unwanted potential presuppositions. (7) is a good example. The implicature from the first clause ('*It is possible that John was not beating the egg.*') cancels the potential presupposition from the second clause ('*John had been beating the egg.*'). However, appealing to individual clauses for the implicatures is what introduces the problem. In sentences such as (10) no clause in the antecedent generates an implicature which is sufficient to cancel the potential presupposition produced by the consequent ('*Mary bought a dress that is the same colour as Susan's.*').

[1]Sentences having more than one verb in their underlying semantic structure. In the surface form this situation is normally exhibited as relative clauses or sentences combined with logical functors (*or, and, if ... then*).

[2]In all of the examples only those presuppositions of importance to the discussion are mentioned.

[3]I am using Gazdar's terminology here because his definition is given in these terms. In the Default Logic setting there are no potential presuppositions. Instead the implicatures are used to provide the contexts in which the presuppositions are computed.

[4]In Gazdar's theory these potential implicatures become implicatures if they are not cancelled by the context in which they are generated.

[5]How this problem is successfully overcome in the Default Logic method is found in [Mercer, 1988a].

The Default Logic method also requires stronger clausal quantity implicatures than those generated from Gazdar's definition. A detailed description of this problem is given in section 6.

[Landman, 1981] attempts to overcome the deficiencies in Gazdar's method by strengthening the clausal quantity implicatures that are generated. The new definition generates not only $\mathsf{P}_S\psi$ and $\mathsf{P}_S\neg\psi$ from every non-entailed clause ψ found in the sentence, but also $\mathsf{P}_S\xi$ and $\mathsf{P}_S\neg\xi$ for every entailment, ξ, of the ψ's. Soames' [1982] argues that not only does this new definition generate implicatures for a sentence which are not justifiable, but also there are cases in which desired presuppositions are cancelled and there are cases (complex conditionals) in which the needed implicatures are still not produced. Soames' solution similarly attempts to repair a Gazdar-like method by increasing the procedure's cancellation abilities. Rather than increasing the strength of the clausal quantity implicatures, he opts instead for a conversational cancellation method based on [Karttunen and Peters, 1979]. In [Mercer, 1987] I argue at some length against this method.

Like [Landman, 1981], I propose that a stronger version of clausal quantity implicatures is needed. In marked contrast to both [Landman, 1981] and [Soames, 1982], the Default Logic approach does not propose a stronger cancellation method that is better able to cancel over-produced potential presuppositions. Rather its production of presuppositions is more conservative. The implicatures help define the contexts in which presuppositions are generated. So, the definition provides the full set of contexts in which the generation process is to take place. The remainder of this paper focusses on the details of the Default Logic method for computing the presuppositions of complex conditionals with special emphasis on the new definition of clausal quantity conditionals. The reader familiar with [Mercer, 1987] can safely proceed to section 6.

5 Using Default Logic to Generate Presuppositions

The Default Logic approach for generating presuppositions is presented in this section with special attention paid to the importance of clausal quantity implicatures and multiple cases. I provide only a brief introduction to this theory of presuppositions, the foundation of which is Default Logic. This theory is partially based upon the ideas of [Wilson, 1975; Kempson, 1975; Gazdar, 1979]. It differs from this previous work at least in its including Default Logic as the fundamental notion in the definition of presupposition.

The discussion that threads itself through the remainder of this paper uses definitions of *bachelor* and *spinster*. A bachelor is an unmarried male adult, and a spinster is an unmarried female adult. In normal contexts it is said that the use of the term *bachelor* presupposes that the individual being referred to is a male adult. Similarly, the use of the term *spinster* presupposes that the individual being referred to is a female adult. There exist contexts in which the use of these terms does not carry all of these presuppositions.

Using default rules provides a context-sensitive method to generate the appropriate presuppositions. Since the generation process is context-sensitive, there are various ways to cancel a presupposition. Firstly, the context can provide information that directly contradicts what would be presupposed by the sentence if the context did not contain the contradictory information. For example '*That person is not a bachelor — he's only five years old.*' is an instance of this form of cancellation. Secondly, given Grice's Maxims for Cooperative Conversation [Grice, 1975], inferences can be generated from certain utterances that indicate that some presupposition normally generated by a presuppositional environment found in the utterance is to be cancelled. For example, '*My cousin is a bachelor or a spinster.*' presupposes neither '*My cousin is male.*' nor '*My cousin is female.*' Grice's maxims indicate that the speaker must be allowing for the possibility of both bachelor-hood and spinster-hood for his cousin. If the sentence presupposes that '*My cousin is male.*' then it would be impossible that my cousin could be a spinster. Similarly, for female. Thus the sentence does not commit the speaker to either of the inferences that would be licensed if the sentence were '*My cousin is a bachelor.*' or '*My cousin is a spinster.*'. Lastly, for technical reasons (such as the uncancelability of entailments by future discourse) entailments are not considered to be presuppositions.

5.1 Model of Communication

A basic model of communication is the transference of information from a knowledge base, which I call the speaker and denote KB_S, to a knowledge base I call the hearer and denote KB_H. Although the model must be quite complex to capture the range of communicative acts possessed by humans, I make some simplifying assumptions for this paper. Firstly, I assume that only true declarative sentences are communicated and that they are asserted, that is, the intent of the speaker is to communicate facts. Secondly, only additions to KB_H will be considered. So, in this restricted setting the speaker intends that the KB_H is to be updated with the logical form of the sentence just uttered. What is the logical form is not completely understood. For the present purpose I use some of Gazdar's proposals. The semantic portion of the meaning of the sentence is represented in some semantic representation language (here I use a standard first order language) as α, say. Since the speaker must know[6] that α is true, this part of the prag-

[6]Some may disagree that a speaker can (cooperatively) communicate only those propositions that he *knows* to be true. I have presented this part of the cooperative communication act exactly as stated in [Gazdar, 1979]. Gazdar takes a *strong* view of cooperative communication, that is, the speaker cannot communicate falsehoods. So, the strong view would require that if the speaker is only communicating beliefs, then the sentence uttered should be "I believe that α." rather than "α.". For those who take a *weak* view of cooperative conversation, that is, the speaker cannot *knowingly* communicate falsehoods, then beliefs can be communicated without the requirement that they be prepended with "I believe that ...". Whether the strong or weak view of

matic portion of the meaning of the utterance is captured as $\mathsf{K}_S\alpha$, where K_S means '*the speaker knows that*'.[7] Occasionally, the utterance also indicates that some parts of the utterance, although not entailed by $\mathsf{K}_S\alpha$, must be possible, as far as the speaker is concerned, otherwise the speaker would have generated a different utterance. Details of these *clausal quantity implicatures* are given in Section 5.2. If **u** is the sentence uttered and α is its semantic representation, then I will use the notation $\mathcal{G}(\mathbf{u})$ to represent $\mathsf{K}_S\alpha$ and the clausal quantity implicatures generated from **u**. Thirdly, Grice's maxims are further captured by taking the logical closure of the KB_H after the proposition has been added. Here, logical closure allows logical relations other than semantic entailment.

If Default Logic proof theory is used as the logical closure method, certain technical problems arise, including generating the complete range of presuppositions from disjunctive representations and the inclusion of modal operators in KB_H. The method to derive the non-modal cases of KB_H is given in Section 5.2. A more complete discussion, including the motivation and justification of this procedure can be found in [Mercer, 1987; Mercer, 1988b].

5.2 Importance of Clausal Implicatures

Because Default Logic proof theory does not display any analogue to the law of the excluded middle (the antecedents of the default rules must be provable and there is no equivalent to the deduction theorem) and because presuppositions do arise from the clauses of complex sentences, some form of analysis by cases is required. Since a statement is provable in a case analysis only if it is provable in *all* cases representing the statement, the generation of the cases is critical. As in the case of a first order theory, too few cases would allow incorrect statements to be proved. In addition because of the context-sensitive nature of default logic, having too many cases or having inappropriately defined cases could prevent the desired statements being proved.

In general the choice of cases must reflect two principles. Since the case analysis is a proof theoretic analogue of the model theoretic law of the excluded middle, each case must *completely determine* the truth values of each of the disjuncts found in the statement to which case analysis is being applied. Also, since the case analysis is justified solely on linguistic grounds (see [Mercer, 1987; Mercer, 1988b]) for further discussion), the cases must reflect this linguistic situation. To justify a case, the possibility of the statement that *distinguishes* the case must be provable from the original default theory. Since none of the modal statements take part in the proofs, they are left out of the cases. An example should clarify these ideas.

5.3 Example

Suppose the sentence 'A or B' is uttered. The updated hearer's knowledge base

$$KB_H \cup \{\mathcal{G}(\text{'}A \text{ or } B\text{'})\}$$

would be

$$\{\mathsf{K}_S(A \vee B), \mathsf{P}_S A, \mathsf{P}_S\neg A, \mathsf{P}_S B, \mathsf{P}_S\neg B, \alpha_1, \ldots, \alpha_n, \delta_1, \ldots, \delta_n\}$$

where $\alpha_1, \ldots, \alpha_n$ and $\delta_1, \ldots, \delta_n$ are first order statements and default rules, respectively, representing the hearer's knowledge before the utterance, $\mathsf{P}_S A, \mathsf{P}_S\neg A, \mathsf{P}_S B, \mathsf{P}_S\neg B$ are the clausal quantity implicatures, and $\mathsf{K}_S(A \vee B)$ is the pragmatic information about the semantic representation of the uttered sentence. (Details concerning how these implicatures are generated is given in Section 6.1.) Since $A \wedge \neg B$ and $\neg A \wedge B$ *completely determine* (that is, determine the truth values of *both*) A and B, and since the statements $\mathsf{P}_S(A \wedge \neg B)$ and $\mathsf{P}_S(\neg A \wedge B)$ can be derived, $A \wedge \neg B$ and $\neg A \wedge B$ *distinguish* the two cases. Note that although $\mathsf{P}_S A, \mathsf{P}_S\neg A, \mathsf{P}_S B, \mathsf{P}_S\neg B$ are all derivable, none of A, $\neg A$, B, $\neg B$ are candidates for distinguishing a case because, individually, none of them completely determine the truth values of both A and B.

Hence the two cases of the original theory, $KB_H \cup \{\mathcal{G}(\text{'}A \text{ or } B\text{'}\}$, are

$$\Delta_{\text{A or B}_{\textbf{Case1}}} = \{A \wedge \neg B, \alpha_1, \ldots, \alpha_n, \delta_1, \ldots, \delta_n\}$$

$$\Delta_{\text{A or B}_{\textbf{Case2}}} = \{\neg A \wedge B, \alpha_1, \ldots, \alpha_n, \delta_1, \ldots, \delta_n\}$$

As an example of this situation, the sentence '*My cousin is a bachelor or my teacher is a spinster.*' would generate a case in which '*my cousin is a bachelor*' and '*my teacher is not a spinster*' are true and a case in which '*my cousin is not a bachelor*' and '*my teacher is a spinster*' are true. Both cases would contain first order statements providing the definitions of *bachelor* and *spinster* as well as the default rules that generate *male* and *adult* for *bachelor* and *female* and *adult* for *spinster*.

5.4 A Proof-Theoretic Definition of Presupposition

Definition 1 *Let* **u** *be a sentence uttered by a speaker, S, in accordance with Grice's Maxims of Cooperative Conversation. Let KB_H be the hearer's knowledge base before the utterance, and let the default theories $\Delta_{\mathbf{u}_{\mathbf{Case1}}}, \ldots, \Delta_{\mathbf{u}_{\mathbf{Casen}}}$[8] be the first order cases of the theory $KB_H \cup \{\mathcal{G}(\mathbf{u})\}$. A sentence α is a presupposition of* **u** *with respect to KB_H if and only if*

(i) $\Delta_{\mathbf{u}_{\mathbf{Case}i}} \vdash_\Delta \alpha$ and $\alpha \in Th(CONSEQUENTS\{D\})$, for $i = 1, \ldots, n$,

(ii) $KB_H \cup \{\mathcal{G}(\mathbf{u})\} \not\vdash \alpha$,

cooperative conversation is adopted, the logical machinery is basically unchanged. Instead of interpreting K_S in its strong sense, it can be interpreted weakly as '*the speaker knows that he believes that*'. It is noteworthy that this interpretation is similar to but not exactly like "explicit belief" ([Lakemeyer, 1987]). That only explicit beliefs can be intentionally communicated seems a reasonable assumption. So, the only difference between the strong and the weak view of cooperative conversation from a logical/truth perspective is that the strong view requires α to be true in the world whereas the weak view requires α to be true in the speaker's explicit belief space.

[7] P_S means '*for all the speaker knows it is possible that*'.

[8] For purposes of this definition, the only defaults in KB_H are the presupposition-generating defaults. $\vdash_\Delta$ is default derivation, and $Th(CONSEQUENTS\{D\})$ is the deductive closure of the default consequents in KB_H. (Reiter (1980))

(iii) $KB_H \not\vdash_\Delta \alpha$,

(iv) $\Delta u_{\mathbf{Case}i} \not\vdash_\Delta \neg\alpha$, for $i = 1, \ldots, n$.

This definition can be loosely paraphrased as: if α is in the deductive closure of the default consequents and is default-provable but not first-order provable from the utterance, if α is not default-provable from the theory with the utterance removed, and in the case of multiple extension default theories if $\neg\alpha$ is not default-provable (since extensions of normal default theories are orthogonal then α is in all extensions), then α is a presupposition of the utterance.

6 Complex Conditionals

Gazdar [1979] provides a method for generating the clausal quantity implicatures of a sentence $\mathbf{u}$. Basically, for every subsentence, ψ, of the semantic representation α of $\mathbf{u}$, such that ψ and $\neg\psi$ are not entailed (or potentially presupposed) by α, $\mathsf{P}_S\psi$ and $\mathsf{P}_S\neg\psi$ are generated, if consistent with the background knowledge. The example given in Section 5.2 has demonstrated that $\mathsf{P}_S A, \mathsf{P}_S\neg A, \mathsf{P}_S B, \mathsf{P}_S\neg B$ are the clausal quantity implicatures generated from '*A or B*', given that they are consistent with the background knowledge. The same four implicatures are generated from '*if A then B*'. For the presupposition generation procedure presented in section 5, this method for generating the clausal quantity implicatures is fine for these simple sentences. Having $\mathsf{K}_S(A \vee B)$ and $\mathsf{K}_S(A \supset B)$, respectively, is critical for this method. In each of these instances the possibility of the appropriate first order cases ($\mathsf{P}_S(A \wedge \neg B)$ and $\mathsf{P}_S(\neg A \wedge B)$ for '*A or B*' and $\mathsf{P}_S(A \wedge B)$ and $\mathsf{P}_S(\neg A \wedge \neg B)$ for '*if A then B*') can be proved.[9] However, in the complex conditionals, '*if A or B then C*' for instance, $\mathsf{K}_S(A \vee B)$ is lacking, thereby preventing the generation of all of the appropriate cases. What is required is a method for generating the appropriate implicatures in all situations, including the complex conditionals.

The phenomenon that Gazdar has captured as the clausal quantity implicature naturally divides into two subgroups. The subgroup which deals with verbs such as *believe, say,* etc. connect a subject and a proposition. The clausal quantity implicature that Gazdar suggests for these verbs ('*for all the speaker knows the proposition is true*' and '*for all the speaker knows the proposition is false*') is adequate. The second subgroup deals with implicatures generated by *or*, *if ... then ...*, and (surprisingly) *and.* The natural difference between the two subgroups is that the second deals with two propositions whereas the first is concerned with only one. I believe that the weakness displayed by Gazdar's clausal quantity implicatures in this second group is that he focusses on individual propositions and not on the connections between them.

In order to strengthen the implicatures generated by this second group I am forced to give up the common definition for clausal quantity implicatures. However, this split is justified by the naturalness of the two groups displaying this phenomena and by the realization that the stronger implicatures emerge as a result of the involvement of two propositions.

[9]$(\mathsf{K}_S(A \vee B) \wedge \mathsf{P}_S\neg A) \supset \mathsf{P}_S(\neg A \wedge B)$ and $(\mathsf{K}_S(A \vee B) \wedge \mathsf{P}_S\neg B) \supset \mathsf{P}_S(A \wedge \neg B)$ are theorems (see [Chellas, 1980]:123).

6.1 Strengthening Gazdar's Clausal Implicatures

I give two differently motivated methods for generating the desired clausal implicatures for complex conditionals. The outputs for the two methods are the same. The positive aspect of the first method is the uniformity of the clausal implicatures for all three of the logical binary relationships that arise in natural language. The weakness of the first method is that it relies on the use of a scalar quantity implicature to change the '*if ... then*' into an '*if and only if*' (see [Gazdar, 1979] for details). Since I have been attempting in related research to show that the scalar implicatures arise from the clausal implicatures, the need of a scalar implicature to derive the clausal implicatures is definitely an undesirable feature. The second method serves two purposes. Firstly, it indicates that the clausal implicatures found in the first method survive embedding in (at least) the antecedents of complex conditionals. Secondly, it demonstrates a methodology which does not depend on the use of a scalar implicature. It is, however, a non-general method relying solely on the particular structure at hand, that is, the embedding of conjuncts and disjuncts in the antecedent of conditionals.

Method 1. The logical connectives produce the following potential clausal quantity implicatures:

A and B	$\neg\mathsf{K}_S(A \supset B)$
	$\neg\mathsf{K}_S(B \supset A)$
A or B	$\neg\mathsf{K}_S(A \supset B)$
	$\neg\mathsf{K}_S(B \supset A)$
if A then B	$\neg\mathsf{K}_S(\neg A \supset B)$
	$\neg\mathsf{K}_S(\neg B \supset A)$

Of special interest are the following three points: Firstly, the potential clausal quantity implicatures for unembedded *and*'s do not survive because the sentence itself generates the pragmatic information, $\mathsf{K}_S(A \wedge B)$. Secondly, I am equivocal about whether a third implicature is generated for each connective. Also, the implicature for *if A then B* includes negated elements. This I have justified by appealing to the underlying semantic representations of *or* and *if ... then* (see footnote 10). Thirdly, the form of the implicatures that I use below are the equivalent $\mathsf{P}_S(A \wedge \neg B)$ and $\mathsf{P}_S(\neg A \wedge B)$ for *and* and *or* and $\mathsf{P}_S(A \wedge B)$ and $\mathsf{P}_S(\neg A \wedge \neg B)$ for *if ... then.*

The cases used in the presupposition analysis in sections 6.2 and 6.3 are generated according to the following procedure: As in Gazdar's system, each logical connective generates its potential clausal quantity implicatures. The derivability of the cases is as follows:

If A or B then C: Renaming the antecedent P, the *if ... then* connective generates the implicature $\mathsf{P}_S(\neg P \wedge \neg C)$ which is equivalent to $\mathsf{P}_S(\neg A \wedge \neg B \wedge \neg C)$. The embedded *or* produces the two implicatures $\mathsf{P}_S(A \wedge \neg B)$ and $\mathsf{P}_S(\neg A \wedge B)$. These two implicatures, together with the representation of the sentence, $\mathsf{K}_S(A \vee B) \supset C)$, generate the other two cases: $\mathsf{P}_S(\neg A \wedge B \wedge C)$ and $\mathsf{P}_S(A \wedge \neg B \wedge C)$.

If A and B then C: Renaming the antecedent P, the *if ... then* connective generates the implicature $\mathsf{P}_S(P \wedge C)$ which is equivalent to $\mathsf{P}_S(A \wedge B \wedge C)$. The other two cases require the use of a scalar implicature that changes the *if ... then* into an *if and only if.* This change results in the addition of the representation $\mathsf{K}_S(\neg(A \wedge B) \supset \neg C)$ to the sentential representation $\mathsf{K}_S((A \wedge B) \supset C)$. The embedded *and* produces the two implicatures $\mathsf{P}_S(A \wedge \neg B)$ and $\mathsf{P}_S(\neg A \wedge B)$. These two implicatures, together with the added representation for the sentence, $\mathsf{K}_S(\neg(A \wedge B) \supset \neg C)$, generate the other two cases: $\mathsf{P}_S(\neg A \wedge B \wedge \neg C)$ and $\mathsf{P}_S(A \wedge \neg B \wedge \neg C)$.

Similar derivations can be given if the embedded connective is in the consequent of the *if ... then*.

Method 2. In the case of complex conditionals of the form '*if A or B then C*' the following argument can be made: If the speaker knows that A and B are "logically connected", that is, $A \supset B$ or $B \supset A$ then he would have generated a different sentence. For example, '*If Fido is a dog or a mammal then ...*' seems rather strange, whereas '*If Fido is a dog or some other kind of mammal then ...*' seems more acceptable. Another argument exists based on the semantic representation for '*if A or B then C*': $(A \vee B) \supset C$. This representation is equivalent to the two sentences $A \supset C$ and $B \supset C$. If the speaker knows that the two antecedents are logically connected he can simplify his statement. Hence, the hearer can infer that the speaker does not know that the two antecedents are logically connected, that is, $\neg(\mathsf{K}_S(A \supset B) \vee \mathsf{K}_S(B \supset A))$. This is of course equivalent to $\mathsf{P}_S(A \wedge \neg B) \wedge \mathsf{P}_S(\neg A \wedge B)$. Given this result and the representation of the utterance, $\mathsf{K}_S((A \vee B) \supset C)$, $\mathsf{P}_S(A \wedge \neg B \wedge C)$ and $\mathsf{P}_S(\neg A \wedge B \wedge C)$ can be derived. Renaming the antecedent P, the hearer can infer that $\neg\mathsf{K}_S(\neg P \supset C)$ otherwise $\mathsf{K}_S((P \vee \neg P) \supset C)$ which would mean that the speaker could have said '*C.*'. $\neg\mathsf{K}_S(\neg P \supset C)$ is equivalent to the third case $\mathsf{P}_S(\neg A \wedge \neg B \wedge \neg C)$.

Similar arguments can be made for complex conditionals of the form '*if A then if B then C*' (or the equivalent '*if A and B then C*'). Specifically, the hearer can infer that the speaker does not know that one of the antecedents alone implies the consequent. The representation of this inference is $\neg\mathsf{K}_S((A \wedge \neg B) \supset C) \wedge \neg\mathsf{K}_S((\neg A \wedge B) \supset C)$[10] which is equivalent to the cases $\mathsf{P}_S(A \wedge B \wedge C)$ and $\mathsf{P}_S(\neg A \wedge B \wedge \neg C)$. Renaming the antecedent P, the hearer can infer that $\neg\mathsf{K}_S(C \supset \neg P)$ otherwise the speaker could have said '*not P.*'. $\neg\mathsf{K}_S(C \supset \neg P)$ is equivalent to the third case $\mathsf{P}_S(A \wedge B \wedge C)$.

[10]It also seems reasonable to include here $\neg\mathsf{K}_S((\neg A \wedge \neg B) \supset C)$. I have relegated this particular item to this footnote since it does not match what was suggested in Method 1. I have not included it there for two reasons. Firstly, it would take the form $\neg\mathsf{K}_S(\neg A \supset B)$. Note that it contains a negated clause where the semantic representation contains an unnegated one. I have not convinced myself whether this is justified (unlike the situation for '*if A then B*'). Secondly, inclusion of this particular item has repercussions for generating the scalar implicature from the clausal implicature.

6.2 Example — 'if A or B then C'

Given the results discussed in section 6.1, a natural language sentence with the appropriate structure can be examined. Suppose that a speaker utters (11).

(11) If my cousin is a bachelor or my teacher is a spinster then someone at my party is unmarried.

Here, the three first order cases generated by the method discussed in Section 6.1 are:

Case 1: *my cousin is a bachelor, my teacher is not a spinster,* and *someone at my party is unmarried*

Case 2: *my cousin is not a bachelor, my teacher is a spinster,* and *someone at my party is unmarried*

Case 3: *my cousin is not a bachelor, my teacher is not a spinster,* and *no one at my party is unmarried*

Following the Default Logic method for generating presuppositions and entailments, '*my cousin is male and adult*' and '*my teacher is female and adult*' are provable in all cases and since they require the use of a default rule in at least one of the cases, they are considered presuppositions of (11). A different set of cases could have resulted in missing some of these inferences (for instance, if '*my teacher is not a spinster*' is not in Case 1 then the second presupposition would not have been derived) or wrongly labelling them as entailments (for instance, if Case 2 and 3 were not considered, the first presupposition would have been labelled an entailment).

Suppose that (12) is uttered.

(12) If my cousin is a bachelor or (my cousin is) a spinster, then my cousin will be the life of the party.

Here, the three first order cases generated by the method discussed in Section 6.1 are:

Case 1: *my cousin is a bachelor, my cousin is not a spinster,* and *my cousin will be the life of the party*

Case 2: *my cousin is not a bachelor, my cousin is a spinster,* and *my cousin will be the life of the party*

Case 3: *my cousin is not a bachelor, my cousin is not a spinster,* and *my cousin will not be the life of the party*

Following the Default Logic method for generating presuppositions and entailments, '*my cousin is adult*' is provable in all cases and since it requires the use of a default rule in at least one of the cases, it is considered a presupposition of (12), but neither '*my cousin is male*' nor '*my cousin is female*' is an inference. Since Case 3 produces two extensions, one containing '*my cousin is male*' and one containing '*my cousin is female*', the definition of presupposition as given in section 5.4 considers neither of them as presuppositions. As well, Cases 1 and 2 differ on the sex of '*my cousin*' which would disallow either sex as a presupposition of the sentence.

6.3 Example — 'if A then if B then C'

Given the results from section 6.1, a natural language sentence with the appropriate structure can be examined. Suppose that a speaker utters (13).

(13) If my cousin is a bachelor then if my teacher is a spinster then the village matchmaker will be delighted.

Here, the four first order cases generated by the method discussed in Section 6.1 are:

Case 1: *my cousin is a bachelor, my teacher is a spinster,* and *the village matchmaker will be delighted*

Case 2: *my cousin is not a bachelor, my teacher is a spinster,* and *the village matchmaker will not be delighted*

Case 3: *my cousin is a bachelor, my teacher is not a spinster,* and *the village matchmaker will not be delighted*

Case 4: *my cousin is not a bachelor, my teacher is not a spinster,* and *the village matchmaker will not be delighted*

Following the Default Logic method for generating presuppositions and entailments, '*my cousin is male and adult*' and '*my teacher is female and adult*' are provable in all cases and since they require the use of a default rule in at least one of the cases, they are considered presuppositions of (13).

Suppose (14) is uttered.

(14) If my cousin is a bachelor and my teacher is a spinster then the village matchmaker will be delighted.

(14) is considered to be semantically and pragmatically equivalent (at least as the clausal quantity implicatures and presuppositions are concerned) to (13). Refer to the discussion following (13) for the case analysis and presupposition analysis.

6.4 A Generalization of Gazdar's Clausal Quantity Implicature Definition

It is of some interest to note that not only does the new definition of clausal quantity implicatures generate stronger implicatures, but also Gazdar's original definition is a logical consequence of this new definition. It is quite easy to see this generalization: In the case of '*A or B.*' the new implicatures are $\neg \mathsf{K}_S(A \supset B)$ which is equivalent to $\mathsf{P}_S(A \wedge \neg B)$ and $\neg \mathsf{K}_S(B \supset A)$ which is equivalent to $\mathsf{P}_S(B \wedge \neg A)$, from which $\mathsf{P}_S A$ and $\mathsf{P}_S \neg B$, and $\mathsf{P}_S B$ and $\mathsf{P}_S \neg A$ can be derived, respectively. Likewise, for '*if A then B.*' the new implicatures are $\neg \mathsf{K}_S(\neg A \supset B)$ which is equivalent to $\mathsf{P}_S(\neg A \wedge \neg B)$ and $\neg \mathsf{K}_S(B \supset \neg A)$ which is equivalent to $\mathsf{P}_S(B \wedge \neg A)$, from which $\mathsf{P}_S \neg A$ and $\mathsf{P}_S \neg B$, and $\mathsf{P}_S B$ and $\mathsf{P}_S A$ can be derived, respectively.

7 Conclusions

In this paper I demonstrate the application of Default Logic to the generation of natural language presuppositions for the particular class of natural language sentences called complex conditionals. In order that this method for generating natural language presuppositions can be applied to complex conditionals, the notion of clausal quantity implicature requires strengthening. Gazdar [1979] provides a simple method for generating clausal quantity implicatures from a broad class of situations. [Soames, 1982] points out that this version is too weak to obtain the appropriate presuppositions in complex conditionals. The strengthened version proposed here trades the simplicity for a broader scope of applicability. The Default Logic method for computing the presuppositions, previously described in [Mercer, 1987; Mercer, 1988b], remains unchanged.

Acknowledgements

I wish to thank the anonymous referees for their helpful suggestions.

References

[Chellas, 1980] B. F. Chellas. *Modal Logic: An Introduction.* Cambridge University Press, 1980.

[Gazdar, 1979] G. J. M. Gazdar. *Pragmatics: Implicature, Presupposition, and Logical Form.* Academic Press, 1979.

[Grice, 1975] H. P. Grice. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and Semantics, v.3, Speech Acts,* pages 41–58. Academic Press, 1975.

[Karttunen and Peters, 1979] L. Karttunen and S. Peters. Conventional implicature. In C-K. Oh and D. A. Dineen, editors, *Syntax and Semantics, v.11, Presuppositions,* pages 1–56. Academic Press, 1979.

[Kempson, 1975] R. M. Kempson. *Presupposition and the Delimitation of Semantics.* Cambridge University Press, 1975.

[Lakemeyer, 1987] G. Lakemeyer. Tractable meta-reasoning in propositional logics of belief. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence,* pages 402–408, 1987.

[Landman, 1981] F. Landman. A note on the projection problem. *Linguistic Inquiry,* 12:467–471, 1981.

[Mercer and Reiter, 1982] R. E. Mercer and R. Reiter. The representation of presuppositions using defaults. In *Proceedings of the 4th Biennial Conference of the CSCSI/SCEIO,* pages 103–107, 1982.

[Mercer, 1987] R. E. Mercer. *A Default Logic Approach to the Derivation of Natural Language Presuppositions.* PhD thesis, Dept. of Computer Science, University of British Columbia, 1987.

[Mercer, 1988a] R. E. Mercer. Solving some persistent presupposition problems. In *Proceedings of the 12th International Conference on Computational Linguistics,* pages 420–425, 1988.

[Mercer, 1988b] R. E. Mercer. Using default logic to derive natural language presuppositions. In *Proceedings of the 7th Biennial Conference of the CSCSI/SCEIO,* pages 14–21, 1988.

[Soames, 1979] S. Soames. A projection problem for speaker presuppositions. *Linguistic Inquiry,* 10:623–666, 1979.

[Soames, 1982] S. Soames. How presuppositions are inherited: A solution to the projection problem. *Linguistic Inquiry,* 13:483–545, 1982.

[Wilson, 1975] D. Wilson. *Presuppositions and Non-Truth-Conditional Semantics.* Academic Press, 1975.

An Implementation of a Reversible Grammar

Ping Peng and **Tomek Strzalkowski** *
Department of Computer Science
New York University
251 Mercer Street
New York, New York 10012, U.S.A.

Abstract

A reversible grammar has been implemented based on an algorithm for automated inversion of a unification parser for natural language into an efficient unification generator. The formalism which we have adopted is Definite Clause Grammar. The inversion algorithm uses an automated goal ordering technique to realize the reversible procedure for parsing and generation.

1 Introduction

In current natural language processing systems, two separate grammars are used for language parsing and language generation. However, researchers have a long standing interest in designing a single grammar to perform language parsing and language synthesis tasks, for computational efficiency and integrity, as well as linguistic elegance and perspicuity. From the linguistic point of view, the same linguistic information as well as knowledge representation and inference can be shared by language understanding and language generation. Most work in both understanding and generation assumes a taxonomy of basic word classes and shares descriptions for the types of constructions that are available in a specific language. This provides the possibility of designing a dictionary and a grammar to serve both understanding and generation. From the computational point of view, the direction of parsing is the opposite of the direction of synthesizing. The process of parsing differs from the process of synthesizing in its control focus. It is not trivial to implement a natural language processing system with a reversible grammar for efficient understanding and generation.

A method to obtain a reversible logic grammar has recently been proposed by Dymetman and Isabelle [Dymetman and Isabelle, 1988]. The grammar, however, is compiled into a PROLOG parser and a PROLOG generator semi-automatically, since it requires that nonterminals be manually annotated for the order in which they will be expanded during parsing and generation. We have implemented a reversible grammar which uses the inversion algorithm described in [Strzalkowski, 1989] to derive a unification generator from a unification parser automatically.

*This paper is based upon work supported by the Defense Advanced Research Projects Agency under Contract N00014-85-K-0163 from the Office of Naval Research and grant IRI-8902304 from the National Science Foundation.

2 Grammar Formalism

The overall approach embodied in the reversible grammar rules is based on linguistic string grammar [Sager, 1981] and the operator-argument framework [Kosaka *et al.*, 1988]. It provides a well-developed, broad coverage of grammatical constructions, exposes the semantic information structure of the domain, and retains the transformational derivation produced by analyzing the structure of a sentence, such as nominalization, passive, relative clauses, etc. in operator-argument trees produced by the grammar. This information is very significant in helping to synthesize target surface sentences. Linguistically, operator-argument grammar can be used for both language understanding and generation.

Logic programming provides a possible environment for reversible computation [Sterling and Shapiro, 1986]. For example in PROLOG, the predicate `concatenate(X,Y,Z)` serves not only to concatenate two lists X and Y together into Z, but also to take a list Z apart into two sublists X and Y. To capture the idea of which arguments are being computed on the basis of which others, the notion of a *mode* for a PROLOG predicate can be defined [Shoham and McDermott, 1984]. A *mode* tells which arguments are used as inputs and which as outputs. In other words, it specifies which arguments on execution of the predicate are non-variables and which are variables. By reversibility of a PROLOG program, we mean that the same program can be used in different modes. Obviously, there are restrictions on the reversibility of PROLOG programs. We discuss them later in this paper.

In a PROLOG program for a language grammar system, we are interested in its *parsing mode* and *synthesis mode*. To program a PROLOG parser, we axiomatize the context free grammar rules and language restriction rules of operator-argument grammar in definite clauses [Grishman, 1986], [Pereira and Shieber, 1987] & [Shieber, 1986]. The PROLOG proof procedure, in conventional parsing mode, gives a top-down, depth-first, left-to-right parsing mechanism. It takes an instantiated natural language string as an input and produces a parse tree structure as output. In the synthesis mode, the PROLOG proof procedure works as a generator under the same axiomatization of grammar rules and restriction rules. Given an instantiated parse tree structure, it produces a corresponding natural language surface string.

The following is a grammar rule which defines the declarative sentence:

```
assertion(S) -- >
  sa(S1),
  subject(Subj),
  sa(S2),
  verb(V),
  {Subj:np:number ::  V:number}
  sa(S3),
  object(O,V,Vp,Subj,Sp),
  sa(S4),
  {S:verb:head ::  Vp:head},
  {S:verb:number ::  V:number},
  {S:tense ::  [V:tense, O:tense]},
  {S:subject ::  Sp},
  {S:object ::  O:core},
  {S:sa ::  [S1:sa, S2:sa, S3:sa, O:sa, S4:sa]}.
```

Here, :: is a user defined infix operator which succeeds whenever its two arguments can be unified. Usually, one of them is bound in execution. The notation `Var:attr`, where : is another infix operator, should be read as the value of attribute `attr` of structure `Var`. Each literal not surrounded by braces in the above clause represents a context free component in the `ASSERTION` rule in the grammar. The literal `{Subj:np:number::V:number}` states the number agreement restriction. The rest of the literals surrounded by braces are used to construct the regularized parse tree.

Each predicate symbol in a definite clause stands for some linguistic object. Its arguments contain information of different types such as syntactic structures, semantic structures and language restrictions, etc. The syntactic and semantic structures of a linguistic object are built from the syntactic and semantic structures of its children in the derivation tree. The syntax and semantics compositionality supports the grammar reversibility. In parsing mode, a linguistic object is decomposed into the syntactic substructures of its children and its semantics is composed from semantics of its children. In synthesis mode, the semantics of a linguistic object is decomposed into smaller semantic structures of its children, and its syntax is build up from the syntax of its children.

The definite clause grammar is first translated into its equivalent PROLOG program. The grammar rule for `assertion` is translated into the following clause.

```
assertion(S,L1,L2) :-
  sa(S1,L1,L3),
  subject(Subj,L3,L4),
  sa(S2,L4,L5),
  verb(V,L5,L6),
  Subj:np:number ::  V:number,
  sa(S3,L6,L7),
  object(O,V,Vp,Subj,Sp,L7,L8),
  sa(S4,L8,L9),
  S:verb:head ::  Vp:head,
  S:verb:number ::  V:number,
  S:tense ::  [V:tense, O:tense],
  S:subject ::  Sp,
  S:object ::  O:core,
  S:sa ::  [S1:sa, S2:sa, S3:sa, O:sa, S4:sa].
```

In parsing mode, the argument `L1` which is the input language string is instantiated. An invocation of the goal `assertion(S,L1,L2)`, if it is successful, results in the argument `S` being bound to the regularized operator-argument structure of the parsed language string. One would expect that the invocation of the goal `assertion(S,L1,L2)` with the instantiated argument `S` will return `L1` bound to a language string corresponding to the regularized syntactic tree in the synthesis mode. However, the program is most likely to go into infinite loop and be aborted. The problem is that the goals `subject(Subj,L3,L4)`, `verb(V,L5,L6)`, and `object(O,V,Vp,Subj,Sp,L7,L8)`, etc. on the right hand side of the clause are invoked with unbound arguments.

The PROLOG program, which consists of a list of definite clauses and a goal, defines an AND-OR computation tree. In the parsing mode, the PROLOG interpreter traverses this tree depth-first from left to right. The related structures and features are passed along the tree traversal top-down from left to right. In its synthesis mode, the related structures and features need to be passed bottom-up, and often from right to left, while the PROLOG interpreter preserves the depth-first and left-to-right strategy. Thus expanding some nodes with unbound arguments in the tree can create infinite paths, even though these arguments could be bound to features passed from their right siblings. If we adjust the tree traversal strategy to the direction of feature passing in synthesis mode so that each node in the tree is fired with all necessary arguments instantiated, the infinite paths can be eliminated. For a PROLOG program, to change the traversal strategy in its computation is to rearrange the order of goals in the right hand sides of its clauses [1].

3 An inversion algorithm

One idea for capturing the goal ordering is to use the concept of essential arguments [Strzalkowski, 1989]. Essential arguments create a subset of arguments of every literal such that the literal cannot be executed successfully unless all of them are bound, at least partially, at the time of execution of the literal. A minimal set of essential arguments for a literal is defined as a subset of its arguments which are essential and no proper subset of these arguments is essential. The minimal set of essential arguments for a literal varies depending upon the direction of the computation. An active minimal set of essential arguments of a head literal of a clause is defined as its minimal set of essential arguments which need to be bound when this clause is fired under its present computation mode. On the other hand, because of the specific control structure of a PROLOG program, a clause can be executed successfully only if all the literals on

[1] In some situations it may be necessary to move certain literals from one clause to another. We do not discuss this problem in the present paper.

its right hand side are executed successfully in the order they occur, from left to right. If we know the active minimal sets of essential arguments of all right hand side literals for a required computation mode of a clause, we can manage to reorder the literals on the right hand side to execute the clause successfully.

3.1 Essential arguments

In our system, we are interested in the parsing mode and the synthesis mode of a set of PROLOG clauses created for our natural language grammar. For example, consider the following grammar rule for a present participle and an object following the verb *be* in a progressive tense construction:

```
object(O,V,Vp,S,Sp) -- >
  {V:root :: be},
  vingo(O1,S,Sp),
  {O:tense :: [progressive, O1:tense]},
  {O:core :: O1:core},
  {O:sa :: O1:sa},
  {Vp:head :: O1:head}.
```

When this rule is translated into a PROLOG clause by an ordinary DCG interpreter, the PROLOG clause preserves the order of literals in the grammar rule:

```
object(O,V,Vp,S,Sp,L1,L2) :-
  V:root :: be,
  vingo(O1,S,Sp,L1,L2),
  O:tense :: [progressive, O1:tense],
  O:core :: O1:core,
  O:sa :: O1:sa,
  Vp:head :: O1:head.
```

In the parsing mode, the active minimal set of essential arguments of `object(O,V,Vp,S,Sp,L1,L2)` is {`S,L1`}; here `L1` is the current input string for parsing, and arguments `V` and `S` are passed from the clause that invoked `object`. `V` is a structured component in the parse tree for the verb which precedes the object string in the input sentence. It is passed down to `object` for two purposes. The first purpose is to check the subcategorization of the verb which precedes the object. The second purpose is to extract the predicate of the sentence to build a regularized parse tree. `S` is a structured component in the parse tree representing the subject of the sentence. It is passed down to `object` to facilitate handling of passive and embedded sentences. The active minimal set of essential arguments of `V:root::be` is an empty set. `be` is a constant. When `V` is bound, the predicate checks the equality of the two sides. When `V` is not bound, `V:root` will be bound by the constant `be`. The active minimal set of essential arguments of `vingo(O1,S,Sp,L1,L2)` is {`S,L1`}, where `L1` is the input string for parsing, and `S` is the structured component as described above. The active minimal sets of essential arguments of `O:tense::[progressive, O1:tense]`, `O:core::O1:core`, `O:sa::O1:sa`, and `Vp:head::O1:head` are {`O1`} and {`O`}. But only {`O1`} is bound in parsing mode. `O1` is the structured component built in `vingo(O1,S,Sp,L1,L2)`. It will become a part of the structured component `O`. Having these active minimal sets of essential arguments for each literal in the clause, we examine the execution of the clause for parsing. To complete the execution of `object(O,V,Vp,S,Sp,L1,L2)`, the left most literal on the right hand side is fired first. The active minimal set of essential arguments of the first goal `V:root::be` is an empty set, thus it can be executed successfully. The literal `vingo(O1,S,Sp,L1,L2)` is fired next. The essential arguments `S` and `L1` are bound by features passed by `object`, thus it too can be executed successfully and return `O1` bound. Then `O1`, as the essential argument for the rest of the literals, is used to execute those literals successfully. This completes the execution of `object(O,V,Vp,S,Sp,L1,L2)`, which returns `O,Sp,Vp` and `L2` bound to its caller.

In its synthesis mode, the active minimal set of essential arguments of `object(O,V,Vp,S,Sp,L1,L2)` is {`O,Vp,Sp`}. `O`, `Vp` and `Sp` are the regularized syntactic structures for the object, predicate and subject of the synthesized sentence, respectively. They are used for producing a corresponding sentence string. The active minimal set of essential arguments of `V:root::be` is again an empty set. This literal generates a surface verb *be*, which precedes the predicate of the sentence in its progressive tense. The active minimal set of essential arguments of `vingo(O1,S,Sp)` is {`O1,Sp`}. The literal `vingo(O1,S,Sp)` generates a surface object string following the verb *be*. The active minimal sets of essential arguments of `O:tense::[progressive, O1:tense]`, `O:core::O1:core`, and `O:sa::O1:sa` are {`O`} and {`O1`} as before, but only {`O`} is bound in the synthesis mode. These goals decompose the regularized syntactic structure of the object into several substructures. The active minimal sets of essential arguments of `Vp:head::O1:head` are {`Vp`} and {`O1`}. The argument `Vp` passes the root structure of the sentence predicate down to its children to generate an appropriate surface verb in the sentence string. If we attempt to generate an object string from its semantic structure by firing the clause `object(O,V,Vp,S,Sp,L1,L2)` with the same sequence of execution of right hand side literals as that in parsing, we run into trouble in executing the literal `vingo(O1,S,Sp,L1,L2)`. This is because `O1` is an essential argument to the predicate `vingo`, but it is not bound when `vingo` is fired and the procedure goes into an infinite loop.

If the order of right hand side literals is rearranged such that `vingo(O1,S,Sp,L1,L2)` is fired after `O1` is bound, the goal `vingo` can be executed successfully. The adjusted right hand side of the clause may take the following order:

```
object(O,V,Vp,S,Sp,L1,L2) :-
  V:root :: be,
  O:tense :: [progressive, O1:tense],
  O:core :: O1:core,
  O:sa :: O1:sa,
  Vp:head :: O1:head,
  vingo(O1,S,Sp,L1,L2).
```

3.2 Goal reordering algorithm

When attempting to expand a literal on the rhs of any clause the following basic rule should be observed: never expand a literal before at least one its active set of essential arguments has all its elements bound. The following procedure *INVERSE* uses this simple principle to reorder rhs of parser clauses for reversed use in generation. This procedure uses the information about input and output arguments (denoted as "in" and "out", respectively), and essential arguments for predicates, which are computed by separate procedures. The arguments of *INVERSE* are: *clause* which has the form of *head :- old-rhs, ins* which is the set of these variables in *head* that are known to be bound whenever *clause* is executed, and *outs* which is the set of arguments in *head* that are required to be bound after the clause execution is completed. The procedure is slightly simplified here, but see [Strzalkowski, 1989] for more details.

```
INVERSE("head :- old-rhs",ins,outs);
begin
  compute M the set of all sets of essential arguments for
  head;
  for every m ∈ M do begin
     OUT := ∅ ;
     if m is active and m ⊆ ins then begin
       compute and mark "out" arguments in head;
       add them to OUT;
       if outs ⊆ OUT then DONE("head:-old-rhs")
     end
     else if m is non-active and m ⊆ ins then
       begin
       new-rhs := ∅; old-rhs-1 := old-rhs; QUIT := false;
       repeat
         mark "in" those arguments in old-rhs-1 which are
                either "in" in head,
                or "in" or "out" in new-rhs;
         select a literal L in old-rhs-1 with a bound active
                set $m_L$ of ess. args.;
         set up a backtracking point with the remaining
                alternatives to select L;
         if L exists then begin
           if $m_L$ is non-active in L then
             forevery clause "L1 :- $rhs_{L1}$" in the program
               such that L1 has the same predicate as L do
                 begin
                 INVERSE("L1 :- $rhs_{L1}$", $M_L$, ∅);
                 if GIVEUP returned
                 then backup to the latest backtracking point
                 end;
           compute and mark "in" and "out" arguments in L;
           add "out" arguments to OUT;
           new-rhs := APPEND-AT-THE-END(new-rhs,L);
           old-rhs-1 := REMOVE(old-rhs-1,L)
         end if
         else begin
           backup to the latest backtracking point;
           if no such backtracking point exists
           then QUIT := true
         end else
       until old-rhs-1 = ∅ or QUIT;
       if outs ⊆ OUT and not QUIT
       then DONE("head:-new-rhs")
     end elseif
  end; for
  GIVEUP("program cannot be inverted as specified")
end;
```

4 The Implementation

Using the concept of the active minimal set of essential arguments in a certain computation mode and the algorithm for ordering literals on the right hand side of a clause, we have implemented an interpreter, which translates Definite Clause Grammar dually into a PROLOG parser and a PROLOG generator. The interpreter first translates Definite Clause Grammar rules into their equivalent PROLOG clauses, which work as a natural language parser. The parser takes a natural language sentence as its input and produces a regularized parse trees. Then the interpreter rearranges the order of literals on the right hand sides of parser clauses and thus inverts the parser into a generator. The generator takes a regularized parse tree as an input and produces a natural language string.

For example, we have the following grammar rule:

```
assertion(S) -->
  sa(S1),
  subject(subj),
  sa(S2),
  verb(V),
  {Subj:np:number ::  V:number},
  sa(S3),
  object(O,V,Vp,Subj,Sp),
  sa(S4),
  {S:verb:head ::  Vp:head},
  {S:verb:number ::  V:number},
  {S:tense ::  [V:tense, O:tense]},
  {S:subject ::  Sp},
  {S:object ::  O:core},
  {S:sa ::  [S1:sa, S2:sa, S3:sa, O:sa, S4:sa]}.
```

When translated into its PROLOG equivalent, it yields the following clause in the parser:

```
assertion(S,L1,L2) :-
  sa(S1,L1,L3),
  subject(Subj,L3,L4),
  sa(S2,L4,L5),
  verb(V,L5,L6),
  Subj:np:number ::  V:number,
  sa(S3,L6,L7),
  object(O,V,Vp,Subj,Sp,L7,L8),
  sa(S4,L8,L9),
  S:verb:head ::  Vp:head,
  S:verb:number ::  V:number,
  S:tense ::  [V:tense, O:tense],
  S:subject ::  Sp,
  S:object ::  O:core,
  S:sa ::  [S1:sa, S2:sa, S3:sa, O:sa, S4:sa].
```

The inverted `assertion` predicate as it appears in the generator is shown below. The prefix g_ is added to each inverted predicate in the generator to distinguish them from their non-inverted version used in the parser.

```
g_assertion(S,L1,L2) :-
  S:verb:head ::  Vp:head,
  S:verb:number ::  V:number,
  S:tense ::  [V:tense, O:tense],
  S:subject ::  Sp,
  S:object ::  O:core,
  S:sa ::  [S1:sa, S2:sa, S3:sa, O:sa, S4:sa],
    g_sa(S4,L3,L2),
    g_object(O,V,Vp,Subj,Sp,L4,L3),
    g_sa(S3,L5,L4),
    Subj:np:number ::  V:number,
    g_verb(V,L6,L5),
    g_sa(S2,L7,L6),
    g_subject(Subj,L8,L7),
    g_sa(S1,L1,L8).
```

4.1 Overall organization

In the process of reversing, we first compute the active minimal sets of essential arguments in the generation mode for each clause. Then we rearrange the order of the right hand side literals for each clause based on the obtained active minimal sets of essential arguments of each literal within the clause. To implement the algorithm efficiently, we actually process the computation of the minimal sets of essential arguments and the reordering of the right hand size literals in one pass in the process that interprets the DCG grammar.

We consider each clause in the grammar as an ordered tree. The head of the clause is placed at the root of its ordered tree and the right hand side literals are placed at the leaves of the tree. For each clause tree the minimal set of essential arguments of the root is determined from the minimal sets of essential arguments of all its leaves and the way they are ordered. The terminals in a grammar are connected directly to dictionary entries. Initially, we assume that the minimal sets of essential arguments of these dictionary access and other primitives are known. A queue is maintained to keep all literals which have their minimal sets of essential arguments known. For each literal in the queue, its parent root is found and informed about the available minimal set of essential arguments of the child. As soon as the minimal sets of essential arguments of all the leaves in a clause tree are known, the minimal set of essential argument of the root is computed. If it conforms to the requirements of the synthesis mode, the goals on the right hand side are reordered accordingly. Since the head of a clause can appear on the right hand sides of other clauses in the grammar, the minimal set of essential arguments of the root of one clause can serve as the minimal set of essential arguments of a leaf of another clause. The minimal sets of essential arguments of all clauses are obtained in the same way. If a grammar consists of all productions that derive strings of terminals and are derivable from the start symbol, the computation is complete [Cai and Paige, 1988/89] and the sets of essential arguments for all clauses are obtained.

The following is one of the grammar rules cited earlier:

```
object(O,V,Vp,S,Sp) -- >
  {V:root ::  be},
  vingo(O1,S,Sp),
  {O:tense ::  [progressive, O1:tense]},
  {O:core ::  O1:core},
  {O:sa ::  O1:sa},
  {Vp:head ::  O1:head}.
```

The equivalent PROLOG clause is:

```
object(O,V,Vp,S,Sp,L1,L2) :-
  V:root ::  be,
  vingo(O1,S,Sp,L1,L2),
  O:tense ::  [progressive, O1:tense],
  O:core ::  O1:core,
  O:sa ::  O1:sa,
  Vp:head ::  O1:head.
```

The head of this clause is `object`. It is placed at the root of the clause tree. Its children are all the literals on the right hand side of the clause. The children are placed at the leaves of the tree. Suppose we know the minimal sets of essential arguments of all the children of the ordered clause tree at this point, we want to find the minimal set of essential arguments of the root. The literal leaves are examined from left to right. The predicate `V:root::be` is examined first. This is a predicate to assign values from the argument on one side to the other. In this case, one side is a constant. The predicate can be executed successfully at any time. We put it into the output queue. The literal `vingo(O1,S,Sp)` is examined next. {`O1,Sp`} is its minimal set of essential arguments in the synthesis mode. Since it is not a subset of the set of variables occurring at the root, it may not be fully bound for the execution. Therefore it is put into the waiting stack. Following `vingo(O1,S,Sp,L1,L2)`, there is a literal `O:tense::[progressive,O1:tense]`, where {`O`} is the minimal set of essential arguments. It is also a subset of the set of variables at the root. We also find that `O` is not an output variable of any other literals to its right. The variable `O` at the root is therefore added to the minimal set of essential arguments for `object`. The literal `O:tense::[progressive,O1:tense]` is put into output queue, right after `V:root::be` which is already in the queue. A similar analysis is applied to the rest of the literals. This procedure is subsequently repeated for all the literals in the waiting stack until all the literals are output. As a result, the `object` clause above is inverted into a generator clause by rearranging the order of the literals on its right hand side as shown below:

```
g_object(O,V,Vp,S,Sp,L1,L2) :-
   V:root ::  be,
   O:tense ::  [progressive, O1:tense],
   O:core ::  O1:core,
   O:sa ::  O1:sa,
   Vp:head ::  O1:head,
   g_vingo(O1,S,Sp,L1,L2).
```

When there is more than one clause having the same predicate at its head literal, the minimal set of essential arguments for the predicate is computed as the union of the minimal sets of essential arguments obtained for each instance.

4.2 Recursive clauses

In the process of grammar inversion, we may encounter deadlock-like situations when recursive goals are involved. For example, consider the grammar rules for `object` and `vingo` shown below:

```
object(O,V,Vp,S,Sp) -- >
  {V:root ::  be},
  vingo(O1,S,Sp),
      {O:tense ::  [progressive, O1:tense]},
      {O:core ::  O1:core},
      {O:sa ::  O1:sa},
      {Vp:head ::  O1:head}.

   vingo(O,S,Sp) -- >
      ving(V),
      sa(S1),
      object(O1,V,Vp,S,Sp),
      sa(S2),
      {O:head ::  Vp:head},
      {O:tense ::  O1:tense},
      {O:core ::  O1:core},
      {O:sa ::  [S1:sa, O1:sa, S2:sa]},
```

To compute the minimal set of essential arguments for `object`, it is necessary to know the minimal set of essential arguments for `vingo`. On the other hand, to compute the minimal set of essential arguments for `vingo`, the minimal set of essential arguments for `object` has to be known first. We can break this deadlock by using a partially computed minimal set of essential arguments of one of these to compute the partial minimal set of essential arguments of the other. In the above example, the partial minimal set of essential arguments for `object` is obtained from its non-recursive clauses. This set is used for computing the partial minimal set of essential arguments for `vingo`. The partial minimal set of essential arguments for `vingo` is further used for completing the computation of minimal set of essential arguments for `object`. If the complete minimal set of essential arguments for `object` is a super-set of the partial minimal set of essential arguments for `object`, the computation of the minimal set of essential arguments for `vingo` has to be redone by replacing the partial minimal set of essential arguments for `object` with the complete minimal set of essential arguments for `object` in the process. This step is repeated for each recursive clauses until all minimal sets of essential arguments need no further expansion.

4.3 An example

A single grammar is loaded and translated into a PROLOG parser and PROLOG generator through our interpreter. We define the predicate `parse_generate(S,P)` to show the reversible computation. Either the parser or the generator is invoked depending upon the binding status of its arguments. If `S` is bound, the parser is invoked. If `P` is bound the generator is invoked.

```
| ?- load_grammar(grammar).

yes
| ?- parse_generate([the,students,are,taught,by,joe],
|                    P).


P = [ [cat|assertion],
      [tense,present,passive],
      [verb|teach],
      [subject,[np,[head|joe],
                   [number|singular],
                   [class|nstudent],
                   [tpos],
                   [apos],
                   [modifier,null]]],
      [object,[np, [head|student],
                   [number|plural],
                   [class|nstudent],
                   [tpos|the],
                   [apos],
                   [modifier,null]]],
      [sa]];

no

| ?- parse_generate(S,
|                  [[cat|assertion],
|                   [tense,present,passive],
|                   [verb|teach],
|                   [subject,[np,[head|joe],
|                               [number|singular],
|                               [class|nstudent],
|                               [tpos],
|                               [apos],
|                               [modifier,null]]],
|                   [object,[np,[head|student],
|                               [number|plural],
|                               [class|nstudent],
|                               [tpos|the],
|                               [apos],
|                               [modifier,null]]],
|                   [sa]]).
S = [the,students,are,taught,by,joe]

yes
```

5 Conclusions

We have implemented a reversible grammar based on the algorithm for automated inversion of a unification parser for natural language into an efficient unification generator. The declarative content of the grammar is shared by both the parser and the generator. The grammar is compiled dually into a parser and the corresponding generator automatically. We presented a procedure for reversing a grammar based on the operator-argument syntactic structure level. The grammar inversion procedure described here can be further augmented to allow for inter-clausal ordering of goals [Strzalkowski, 1990], that will allow for static inversion of a wider class of grammars.

Acknowledgements

We would like to thank Professor Ralph Grishman and other members of PROTEUS group for their helpful discussion and valuable comments to our work.

References

[Cai and Paige, 1988/89] Jiazhen Cai and Robert Paige. Program derivation by fixed point computation. *Science of Computer Programming 11 (1988/89) 197-261*, 1988/89.

[Dymetman and Isabelle, 1988] Marc Dymetman and Pierre Isabelle. Reversible logic grammar for machine translation. *The Proceedings of the Second International Conference on Theoretical and Method-*

ological Issues in Machine Translation of Natural Languages, 1988.

[Grishman, 1986] Ralph Grishman. *Computational Linguistics.* Cambridge University Press, 1986.

[Kosaka *et al.*, 1988] Michiko Kosaka, Virginia Teller, and Ralph Grishman. A sublanguage approach to japanese-english machine translation. *Proceedings of the Conference on New Directions in Machine Translation*, 1988.

[Pereira and Shieber, 1987] Fernando Pereira and Stuart Shieber. *Prolog And Natural Language Analysis.* Center for the Study of Language and Information, 1987.

[Sager, 1981] Naomi Sager. *Natural Language Information Processing.* Addison-Wesley, 1981.

[Shieber, 1986] Stuart Shieber. *An Introduction to Unification-Based Approaches to Grammar.* Center for the Study of Language and Information, 1986.

[Shoham and McDermott, 1984] Yoav Shoham and Drew McDermott. Directed relations and inversion of prolog programs. *Proceedings of The International Conference on Fifth Generation Computer Systems*, 1984.

[Sterling and Shapiro, 1986] Leon Sterling and Ehud Shapiro. *The Art of Prolog.* The MIT Press, 1986.

[Strzalkowski, 1989] Tomek Strzalkowski. Automated inversion of a unification parser into a unification generator. *PROTEUS project Memorandum#25*, 1989.

[Strzalkowski, 1990] Tomek Strzalkowski. How to invert a parser into an efficient generator: an algorithn for logic grammars. *To appear in Proceedings of 13th COLING*, 1990.

Modelling semantic flexibility with a structured connectionist implementatation of functional category organization[*]

Susan Hollbach Weber
International Computer Science Institute, Berkeley CA, USA

Abstract

This paper describes a neurally inspired model of semantic flexibility. According to the principle of semantic flexibility, only contextually relevant semantic features are activated in noun comprehension. This position is not inconsistent with models of parallel semantic priming at lexical access. The variant presented here combines eventual contextual selectivity with initially prototypical (non-contextual) priming at lexical access.

The proposed model is part of a structured connectionist knowledge representation and inferencing system known as DIFICIL. The conceptual representation of concrete nouns in DIFICIL is structured internally as a complex of distinct functionally motivated *aspects*, each of which is a cluster of mutually excitatory feature values. The default aspects form the central or context independent core of the representation, giving rise to direct inferences in discourse understanding and prototype effects at lexical access and in categorization tasks, among other things. Contextual priming of an aspect inhibits retrieval of all features associated with competing aspects, resulting in the delayed response effects predicted by semantic flexibility. At the same time it facilitates retrieval of features both in the primed aspect and in related aspects, giving rise to distinct prototype effects for modified categories.

1 Introduction

A wide variety of documented psychological effects, including direct inferences, adjective-noun modification, and semantic flexibility, can be modeled by assuming that categories are structured internally as a complex of interrelated aspects, each consisting of mutually reinforcing properties. Green apples, for example, are sour, hard and dry, and each of these properties will, once established, reinforce the others. Mutual inhibition can also occur between the scalar values on a property scale: an apple cannot be both crisp and mushy at the same time. As incompatible values participate in distinct aspects, activation of one aspect may indirectly inhibit another. By the same token, if two aspects share one or more property values, they will tend to be mutually reinforcing.

A crucial assumption underlying this work is that categories, as mental constructs of active agents, are inseparably linked with the agent's goals. The functional properties of an objects supply the organizational basis for the aspects, creating the various perspectives from which the category can be viewed. That is, each functional property corresponds to an aspect, and there are explicit interrelationships between aspects dictated by the planning-level constraints on the functional properties (implicit interrelationships arise at the property value level). For example, when the word 'apple' is spoken, a mental image is conjured up of the visual appearance of an apple. If the agent is hungry at the time, the apple's taste and texture might spring to mind. If the agent hopes to plant an orchard, the seeds are of greatest interest. If the agent disapproves of the evening's entertainment, its properties as a handy projectile as well as its softness and juiciness when rotten come to the fore (see Figure 1). Thus while a category appears stable to the agent, since the same basic set of properties and values is being drawn on at all times, the structure is actually dynamic, shaped by context.

Aspects of a category can be established by several mechanisms. The direct approach is to name a property value participating in the aspect. For example, the phrase 'green apple' supplies evidence for the relevance of the *unripe* aspect. A similar effect is achieved by actually naming the aspect (eg. 'unripe apple'), since the motivating functional property value which gives the aspect its name also participates in the coalition. Sometimes one aspect will be subsumed by another, as, for example, when one planning goal is a subgoal of a higher level goal. When this happens, invocation (by whatever means) of the high level goal supplies activation to the subgoal, thus indirectly exciting the subordinate aspect. Property inheritance can also supply indirect activation to an aspect. If a subcategory names a property value

[*]Thanks to Jerry Feldman for comments on a draft. This work was supported by ONR research contract no. N00014-82-K-0193 and U.S. Army Communication-Electronics Command Grant no. DAAB 10-87-K-022, while at the Computer Science Department, University of Rochester, Rochester NY.

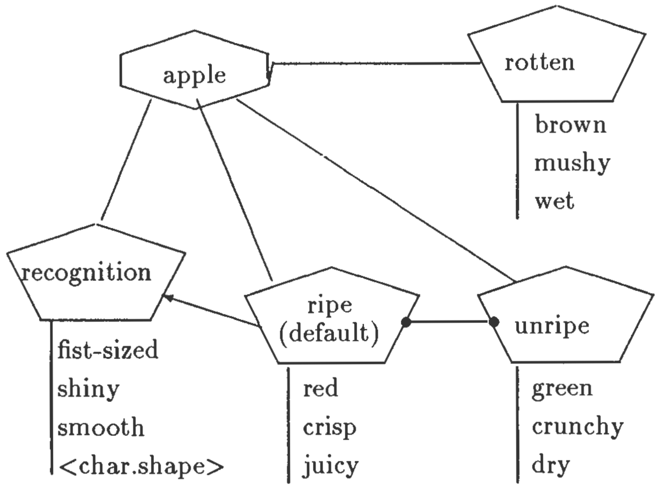


Figure 1: Functional aspects of the *apple* category; aspects appear as pentagons, with participating property values listed below.

that participates in an aspect of the given category, be it the organizational functional value or another, then when activation propagates up from the subcategory to the category in question, that aspect will be preferred. Finally, an aspect can be established by default; in the absence of information to the contrary, the most typical aspect of the category will predominate.

Information to the contrary, that is, evidence against the establishment of an aspect, can also accrue from different sources. One aspect can be explicitly stated to be incompatible with another, in which case mutual inhibition is immediate and unattenuated. A subtler form of inhibition occurs when the two aspects involve two mutually exclusive property values. As only one of the two values can be on at a time, the evidence supporting one aspect will be greater than that of the other. The extreme form of this occurs when one aspect systematically names properties opposed to those in another aspect. The end result of such a situation is the same as explicitly establishing the mutual exclusion between the aspects, only the decision is reached with greater speed and confidence for explicit mutual exclusion.

This model of conceptual representation has been implemented as a structured connectionist network [Feldman and Ballard, 1982] on the Rochester Connectionist Simulator [Goddard *et al.*, 1988], resulting in an inferencing system known as DIFICIL, for Direct Inferences and Figurative Interpretation in a Connectionist Implementation of Language [Weber, 1989b]. Statements in a high level language are translated into connectionist network fragments, permitting a user of the system to create large knowledge bases with relative ease. The *subcat* statement sets up the property inheritance or subcategorization hierarchy. The *mutex* and *invokes* statements specify the relations of mutual incompatibility and reinforcement that pertain between aspects. The ***hasslot*** statement establishes the properties and values belonging to a category, with syntactic variants for perceptual, constitutive and functional properties, and optional scalar positioning parameters. The *aspect* statement creates the conceptual aspects fundamental to direct inferences. For example, the some of the statements used to create the aspects of an apple sketched in Figure 1 are:

hasPslot (apple: color; red, green, brown)
hasFslot (apple: age; rotten (+), ripe (0), unripe (-))
aspect (apple: ripe [default]; crisp, juicy, red)
aspect (apple: unripe; crunchy, dry, green)
mutex (apple: ripe, unripe)
invokes (apple: ripe; recognition)

Actual details of the *aspect* statement appear in Figure 2; the connectionist structures resulting from the four given input statement are depicted with polygons and circles representing connectionist units, and lines and vectors for bi- and uni-directional links respectively. Inhibitory links show up as dots rather than arrowheads. The unit labelled ***hub*** in the diagram is an *inertial binder*, that is, it tends to stay lit once activated (details of how this is accomplished appear in [Weber, 1989b]). It links three other *gated binder nodes*, shown as triangles in the figure, representing the binding between a category, property, and single property value, as established by a *hasslot* statement. Gated binders require input from a distinguished site to be active before summing and broadcasting their input. The distinguished input comes from the associated category (these links are only shown for the *menu-role* property in the figure), reflecting the intuition that a concept–property–value triplet should only be established if one is already considering the category. Note the mutual exclusion enforced between the *calories* value binders, as required by the scalar nature of the property. The default activation on the *food–menu-role–dessert* triplet is established by the *aspect* statement. Thus if the category *food* is under consideration, activation will spread from the *dessert* binder to the hub, and from there to the *rich* and *sweet* binders, establishing that the agent's prototypical food is a dessert. More interestingly, if one establishes that apples are a subcategory of food and that apples are sweet, thus:

subcat (food:apple)
hasPslot (apple: taste; sweet)
aspect (apple: ripe [default]; sweet, red, ...)

then activation will spread not only from *apple* to *sweet*, but also from *apple* to *food*, at which point the conjunction *food–taste–sweet* once again establishes the *dessert* aspect of food, even if this is not a default aspect for food.

A rather interesting technical issue is raised in the implementation of the property inheritance hierarchy. Simple bidirectional links between two categories would create local feedback loops that rapidly raise activation levels of all activated nodes to saturation, thus losing the relative information that is crucial to a meaningful interpretation of the network. Thus the connections between a category and its subordinates are implemented as pairs of mutually exclusive one way flow conduits. In connectionist terms, there are two mutually inhibitory

hasCslot (food: calories; low-cal (-), rich (+))
hasPslot (food: taste; sour, salt, sweet)
hasFslot (food: menu-role; main-course, dessert)
aspect (food; dessert [default]; sweet, rich)

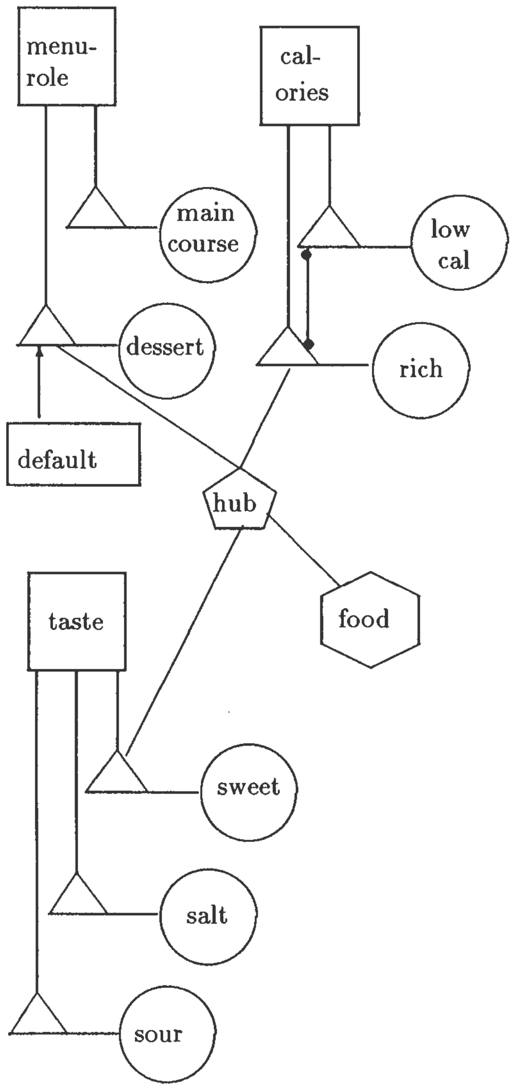


Figure 2: Connectionist structures implementing the DIFICIL model of categories. The uni-directional links from the *food* category node to the three properties *calories, taste* and *menu-role* are not shown.

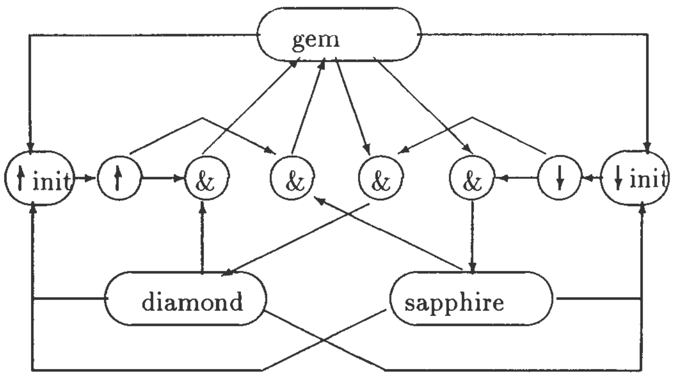


Figure 3: Full depiction of the *subcat* mechanism; the statement used to create this fragment is: *subcat* (gem: diamond, sapphire).

gating nodes, '↑' and '↓', one permitting upward flow of activation, the other downward flow. Distinct gating nodes are associated with each subnet construction, so although activation can flow locally in only one direction, global activation can flow both up and down the hierarchy from the point of origin. These mechanisms also indirectly enforce mutual exclusion amongst the subordinate nodes in the local structure.

There is additional machinery (shown in Figure 3) required to flip activation from ↓ to ↑ and back as needed. The ↑ node is activated (and ↓ inhibited) whenever a subordinate node is active but the parent category is not. Similarly, when the parent is active but none of its subordinates are, the ↓ node becomes active if not already. This behavior is achieved with two additional nodes, called ↑*init* and ↓*init*. Both are linked to the parent category at one site and to all subordinates at another. If only the parent input is active, ↓*init* sends out an excitatory signal to ↓ and an inhibitory one to ↑. Conversely ↑*init* activates ↑ and inhibits ↓ if only the subordinate input is active.

2 Direct inferences

The DIFICIL model of conceptual representation can be used to account for a wide variety of psychological effects. One such effect is the ubiquitous and largely unconscious habit of inferring non-observable properties of objects from perceptual ones, a crucial but widely ignored factor in discourse understanding. These *direct inferences* transform discourse from a string of non-sequiteurs into a logical sequence of ideas. For example, the conversation fragment " 'My apple at lunch was green' 'Here, have a doughnut' " involves several immediate inferences, including some cued by perceptual properties (green apples are sour and inedible), and some about default properties or *prototypes* (apples are typically sweet; they are also the prototypical dessert in a brown bag lunch). Knowledge about frames [Minsky, 1986] (a meal is incomplete without dessert) and categorization (anything sweet can serve as dessert) completes the picture. But without the information that although apples are normally sweet, a green apple is sour, the exchange makes no sense.

Immediate inferences are the direct inferences available at the level of the category under consideration. They are performed quickly, in a few hundred milliseconds, and without conscious thought. These immediate inferences must reflect the structure of stored knowledge, since they are available too quickly and effortlessly to involve any complex form of information retrieval. Specifically, they suggest the use of the spreading activation model of semantic memory. The argument is that the patterns of immediate inferences reflect the structure of connections in the underlying spreading activation model, implemented here as a structured connectionist network.

Mediated inferences are the second form of direct inference, where knowledge about a more abstract category is used to supply the information necessary to understand the discourse. Mediated inferences take somewhat longer to obtain than immediate inferences, as they require chaining up the subcategorization hierarchy created by *subcat* statements.

The time factor in performing direct inferences in DIFICIL is well within the limits prescribed by the psychological data on direct inferences [Anderson, 1983]. It takes 5 time steps for activation to propagate up one layer in the subcategorization hierarchy, and less than 10 time steps for the activation to fully propagate to all relevant property values at that level, so given an inheritance hierarchy of 10 layers, the total time to obtain all direct inferences is less than 100 time steps.

3 Adjective-noun modification

The need for immediate inferences can arise in many different contexts. One such context is the conceptual aspect of relevance is the adjectival modification of nouns. It is generally accepted [Osherson and Smith, 1982; Murphy, 1988; Medin *et al.*, 1987] that the impact of a descriptive adjectival modification of a concrete noun can be too complex to model with either prototype of feature-based models of conceptual representation: green apples are sour and unripe, but green grass is cool and moist, while a green bottle may contain wine.

The DIFICIL model affords an interpretation of adjective effects in terms of the interaction between conceptual aspects. Sometimes an adjective will not supply any connotative information about its noun, that is, the property value it names does not participate in any non-default aspects of the category identified by the noun (eg. green broccoli). In this case the only difference from the interpretation of an unmodified noun is an increase in the salience of the named property and value [Smith *et al.*, 1987]. More often, however, the adjective will operate to select one or more alternative aspects of the category.

Medin [1988] reports that adjectival modification of nouns produces distinct prototype effects from unmodified nouns. He observes that people seem to "take the fact that a spoon is wooden as implying something about the values on other dimensions, such as size and shape" [p. 8]. A typical spoon, for example, is metal, about six or seven inches long, with a flattened handle and a well-cusped bowl. A wooden spoon, however, in addition to being made of wood, is much longer, and typically has a round handle and little cusp to the bowl.

This phenomenon is fundamental to the DIFICIL model of conceptual representation, and is implemented by using the adjective to evoke a conceptual aspect of the category denoted by the noun. Thus the representation of the category *spoon* would include the statement

aspect (spoon; wooden; long, round-handled, shallow, thick)

4 Semantic flexibility

The DIFICIL model of conceptual representation can also be used to explain the phenomenon of *semantic flexibility* observed initially by Barclay *et al.* [1974]. They note that the musical properties of pianos are emphasized in the sentence "The man tuned the piano", while their heaviness is most salient in the sentence "The man tried to lift the piano". Such contextual priming also enhances performance in a cued recall task, a fact which seems to indicate that unemphasized properties are not included in the interpretation of the sentence. They propose that only the contextually relevant semantic features of a noun are activated in its interpretation.

The degree to which unemphasized features participate in a noun's interpretation has been a matter of some controversy. Suggestions range from the proposal of Tabossi [1980; 1986], that emphasized features actually inhibit unemphasized features, to the observations of Medin [1987], that emphasis on a particular feature by adjectival modification of the noun can result in priming on otherwise atypical features, leading to a radically different prototype for the modified concept. This apparent controversy can be resolved by adopting the fine-grained view of the internal structure of conceptual representation proposed by DIFICIL.

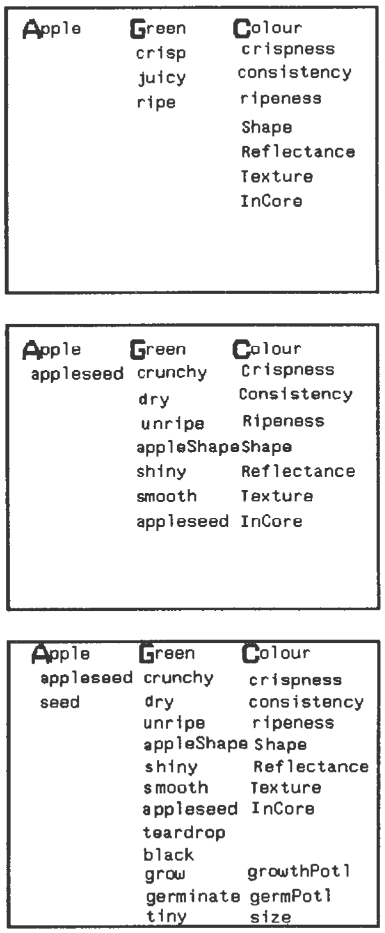
Tabossi finds that lexical access is fastest when primed with a reasonably strong biasing context, slower when in a neutral context, and slowest when a competing feature has been primed. For example, if the sentence primes the 'yellow' feature of gold, the fact that it is malleable is actually inhibited. This is modeled in DIFICIL by the time delay in switching aspects within a category [Hollbach, 1988]. If two aspects are mutually incompatible, it takes significantly longer for the activation in the aspect first primed to be overcome and eventually inhibited by the aspect activated later. For example, when the *ripe* aspect of *apple* has been primed, it takes roughly 50 time steps longer to answer the query "are some apples green?" than when the *unripe* aspect is dominant [Weber, 1989b]. If two aspects are independent of one another, there will be no interference, although Tabossi's results seem to suggest the aspectual competition is the norm. The context shift delay occurs at the level of the aspectual hub, the binder unit relaying activation within the aspect.

Barsalou [1982] proposes a bipartite separation of properties, context independent and context dependent. Context independent properties of a category are activated whenever the category is referred to, while context dependent properties, obviously enough, only crop up in certain contexts. That skunks are smelly and rattlesnakes are poisonous are seemingly invariant properties of their respective categories; they are always thought of no matter what context surrounds mention of the category. The fact that a basketball floats, on the other hand, requires a strongly biasing sentence in order to prime it, such as "Chris used [the basketball] as a life preserver when the boat sank".

The mechanism he offers to explain this behavior is that "properties become automatically activated by a word after being frequently associated with it during processing" [Barsalou, 1982, p. 82]. This notion is incorporated into the DIFICIL model of conceptual representation: context independent properties belong to a default aspect that is compatible with all other aspects of the category, while context dependent properties belong to non-default aspects, some of which may be mutually exclusive.

Greenspan [1986] corroborates Barsalou's findings, reporting that central properties are always instantiated on reference to the noun, while peripheral properties require explicit priming from a biasing context. He also suggests that "emphasized central properties may be *more instantiated* than unemphasized central properties" [p. 544, my italics], though cautioning that this conclusion is not yet supported by experimental results. These relative excitations are displayed in DIFICIL, since the activation supplied to the default features is marginal, while cued activation, eg. from an adjectival modifier, is a full order of magnitude greater in strength [Weber, 1989b].

Greenspan presents further results on the interpretation of an unambiguous word after it has been fully integrated into a sentence context (as opposed to the patterns of initial lexical access of interest to Tabossi). He reports that concepts related to a central property of a mentioned object are activated regardless of the linguistic context [p. 550]. That is, if the word under consideration is *apple*, activation spreads from *apple* to its constitutive property *appleseed*, and from there to the concept of *appleseed*, duly activating all the central properties of seeds. Mechanisms allowing this pattern of activation flow exist in DIFICIL, as direct links exist between the various semantic uses of a given term. Figure 4 demonstrates this capacity for the apple example: the initial stimulus to the network was the phrase 'green apple', and the size of the initial letter of each word is roughly proportional to the degree of activation of the connectionist unit representing it. The top panel shows the default properties of the category 'apple', available 10 time steps after activation was initially keyed in on the input phrase. The center panel, at 20 time steps into the simulation, shows not only the rapid defeating of the default properties of apples, replaced with the characteristic properties of green apples (immediate inferences), but also the spread of activation to the *recognition* aspect and all of its property values. In the bottom panel, the simulation has progressed to the 40th time step, by which time activation has spread from the property *ap-*



```
Apple      Green     Colour
           crisp      crispness
           juicy      consistency
           ripe       ripeness
                      Shape
                      Reflectance
                      Texture
                      InCore
```

```
Apple      Green      Colour
 appleseed crunchy     Crispness
           dry         Consistency
           unripe      Ripeness
           appleShapeShape
           shiny       Reflectance
           smooth      Texture
           appleseed   InCore
```

```
Apple      Green      Colour
 appleseed crunchy     crispness
 seed      dry         consistency
           unripe      ripeness
           appleShape  Shape
           shiny       Reflectance
           smooth      Texture
           appleseed   InCore
           teardrop
           black
           grow        growthPotl
           germinate   germPotl
           tiny        size
```

Figure 4: Time sequence of activation flow from the stimulus words *green apple*.

pleseed to the category *appleseed*, and from there up the subcategorization or abstraction hierarchy to the category *seed*, whose default properties include *size: tiny* and *growth-potential: grow*.

5 Conclusions

As betrayed by the patterns and timing of immediate inferences, the conceptual representation of concrete nouns has an intricate fine-grained structure. Each category is in fact a complex of distinct functionally motivated aspects, which can be excitatory, inhibitory or neutral with respect to each other. The default aspects form the central or context independent core of the representation posited by Barsalou [1982], giving rise to prototype effects in categorization tasks, observed by Rosch[1973]. Priming an aspect will inhibit retrieval of all features associated with competing aspects, resulting in the delayed response effects reported by Tabossi [1986]. It will at the same time facilitate retrieval of features both in the aspect primed and in related aspects, giving rise to distinct prototype effects for modified categories, as noted by Medin [1987]. The functional aspect model of conceptual representation, originally designed to handle immediate inferences, serves as a predictive model for semantic feature retrieval patterns, and suggests a possible line for future experimentation.

References

[Anderson, 1983] John R. Anderson. A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behaviour*, 22:261–295, 1983.

[Barclay *et al.*, 1974] J. R. Barclay, J. P. Bransford, J. J. Franks, N. S. McCarrell, and K. Nitsch. Comprehension and semantic flexibility. *Journal of Verbal Learning and Verbal Behavior*, 13:471–481, 1974.

[Barsalou, 1982] Lawrence W. Barsalou. Context-independent and context-dependent information in concepts. *Memory & Cognition*, 10(1):82–93, 1982.

[Feldman and Ballard, 1982] Jerome A. Feldman and Dana H. Ballard. Connectionist models and their properties. *Cognitive Science*, 6:205–254, 1982.

[Goddard *et al.*, 1988] Nigel Goddard, Kenton Lynne, and Toby Mintz. The rochester connectionist simulator user manual. Technical Report 233, Computer Science Department, University of Rochester, March 1988.

[Greenspan, 1986] Steven L. Greenspan. Semantic flexibility and referential specificity of concrete nouns. *Journal of Memory and Language*, 25:539–557, 1986.

[Hollbach, 1988] Susan C. Hollbach. Direct inferences in a connectionist knowledge structure. *Proceedings of the Tenth Annual Meeting of the Cognitive Science Society*, pages 608–614, 1988. Montreal, Canada.

[Medin and Shoben, 1988] Douglas L. Medin and Edward J. Shoben. Context and structure in conceptual combination. *Cognitive Psychology*, 20, 1988.

[Medin *et al.*, 1987] Doug Medin, Wattenmaker, and Hampson. Family resemblance, conceptual cohesiveness, and category construction. *Cognitive Psychology*, 19:242–279, 1987.

[Minsky, 1986] Marvin Minsky. A framework for representing knowledge. In Ronald Brachman and Hector Levesque, editors, *Readings in Knowledge Representation*. Morgan Kaufman, 1986.

[Murphy, 1988] Gregory L. Murphy. Comprehending complex concepts. *Cognitive Science*, 12:529–562, 1988.

[Osherson and Smith, 1982] Daniel N. Osherson and Edward E. Smith. Gradedness and conceptual combination. *Cognition*, 12:299–318, 1982.

[Rosch, 1973] Eleanor H. Rosch. Natural categories. *Cognitive Psychology*, 4:328–350, 1973.

[Smith *et al.*, 1987] E. Smith, D. Osherson, L. Rips, and M. Keane. Combining prototypes: A modification model. Technical Report 1, Cognitive Science and Machine Intelligence Laboratory, The University of Michigan, January 1987.

[Tabossi and Johnson-Laird, 1980] P. Tabossi and P. N. Johnson-Laird. Linguistic context and the priming of semantic information. *Quarterly Journal of Experimental Psychology*, 32:595–603, 1980.

[Tabossi, 1986] Patrizia Tabossi. Effects of context on the immediate interpretation of unambiguous nouns. Universita di Bologna, Italy. Unpublished manuscript, 1986.

[Weber, 1989a] Susan Hollbach Weber. Figurative adjective-noun interpretation in a structured connectionist network. In *Proceedings of the 11th Annual Conference of the Cognitive Science Society, Ann Arbor, Michigan*, pages 204–211, August 1989.

[Weber, 1989b] Susan Hollbach Weber. A structured connectionist approach to direct inferences and figurative adjective-noun combinations. Technical Report 289, PhD. thesis, Department of Computer Science, University of Rochester, May 1989.

[Weber, 1990] Susan Hollbach Weber. A structured connectionist approach to interpreting figurative adjective-noun combinations. to appear in J. A. Barnden, editor, *Advances in Connectionist and Neural Computation Theory*, volume 2. Norwood, N.J.: Ablex Publishing Corp., 1990.

Route planning by multiple agents

J. L. Weiner and **R. M. Malyankar**
Department of Computer Science
University of New Hampshire
Durham, NH 03824
U. S. A.

Abstract

Multiagent problem solving systems require coordination among their agents. Various techniques have been employed, ranging from game theory to partial global planning, to provide the necessary coordination while at the same time trying to limit communication. Our research focuses on the problem of conflict avoidance and resolution in the context of a system of coordinated and cooperative agents working on a route planning problem. Our proposed agents use a combination of techniques to predict potential conflicts and work towards avoiding them while using limited communication.

1 Introduction

This paper deals with planning by multiple agents in an uncertain environment. Our agents work on a common task and have the same rationality but different knowledge of the environment. This may lead to individual agents making sub-optimal plans if they work on their own. Our aim is to formulate a coordination scheme that allows agents to help each other make a plan that is more optimal than each agent can make on its own, and to do this efficiently. We concentrate on coordination that involves reasoning about another agent's behavior.

Our agents are route planners given the task of planning a route through specified points in the plane in the presence of circular obstacles. Since planning is a kind of search, agents usually have a choice of actions to take at many points during the planning process. Agents work on the same task, so ideally they independently decide on the same choice. But discrepancies in data lead to differences in choices and hence to conflict between agents' ideas of the best actions; this conflict must be resolved so that the agent network can solve the problem faster and more efficiently.

2 Domain

The domain selected was route planning for autonomous vehicles. This domain allows significant cooperation between participating agents. It also allows obvious sources of discrepancies, in different agents' different knowledge about obstacles and other knowledge of the world.

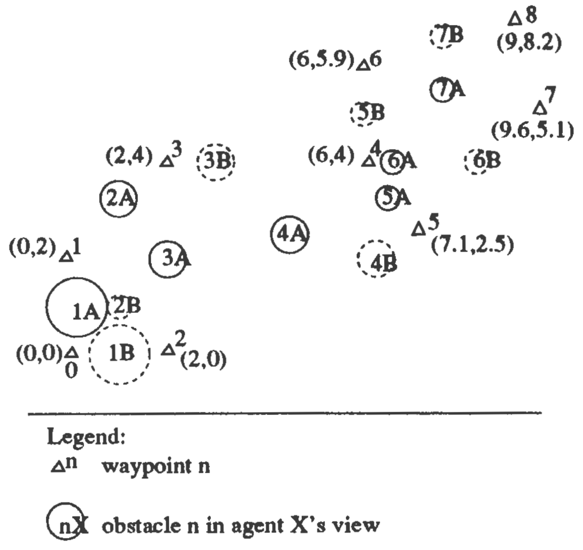


Figure 1. Scene for experiments

To simplify matters, we assume that agents already know the common task. This means that agents need not bother sending or obtaining subproblems. The common goal is a set of tasks that must be carried out at specified places (called waypoints). Our vehicles need to plan a route through all the waypoints mentioned in the prescribed mission. We try to plan the shortest possible path. We restrict ourselves to a single path to be followed by a single vehicle, so that the problem becomes strictly a distributed planning and cooperation problem without the added complication of having to make and maintain a multivehicle plan.

We do not insist on an optimal plan but are content to have a suboptimal plan that is obtained quickly. Insistence on an optimal plan would slow the system but not significantly affect the coordination related behavior since the only effect would be that more goals would be generated and tested.

Figure 1 contains an example of a scene. In this scene, the start is the point (0,0) and the finish is (20,20) (not shown). All obstacles are circular. The text in each circle denotes the obstacle number and the agent which thinks the obstacle is where the circle shows it. The waypoints are laid out so as to present the planner with two almost equally good alternatives at each choice point. The obstacles are placed so as to cause a conflict at each choice point.

Our formulation of the problem is based on the multitask planning problem discussed by B. Hayes-Roth [1985] and the 'mission planning' for an autonomous submersible mentioned by Chappell [1987; 1988].

3 Planning

We concentrate on the cooperation part of the problem; our planner is therefore not very sophisticated, being simply non-backtracking depth-first search. It begins with the start waypoint and finds the nearest waypoints, creating subgoals that consist of joining the current position to each of these. It then plans a path to the nearest waypoint and if this is feasible and agrees with the estimated distance, treats that waypoint as the current point and proceeds as before.

4 Coordination and conflict detection

We assume that agents have the same rationality [Ginsberg, 1987; Rosenschein and Breese, 1988]. We also decide to work on the intermediate goals generated during the planning process; in the domain described, these consist of linking waypoints into successive pairs, with the detailed path between them to be plotted later. There are two problems to be solved: (1) choosing one of the alternatives at a choice point for development, and (2) if agents choose different alternatives, resolving the discrepancy. In a multiagent system, the first is a coordination problem; the second requires a strategy for conflict resolution.

4.1 Coordination schemes

Various coordination and cooperation schemes are possible. Some possible cooperation schemes are the contract net paradigm [Davis and Smith, 1983], the coordinator-coworker hierarchy [Lo and Findler, 1987] and partial global planning [Durfee *et al.*, 1987; Durfee and Lesser, 1987]. All the above are based on agents sending messages to other agents regarding the ordering and distribution of subtasks, the tasks themselves, and data transfers from one agent to another. They do not assume that all agents know everything about another agent and about what that agent knows, though agents have differing degrees and types of knowledge about the systems of which they are members. This knowledge ranges from knowledge about the capabilities, state and tasks allotted each agent to a mere knowledge of their existence. Other coordination schemes such as game theory [Ginsberg, 1987], depend on complete knowledge about other agents. This scheme needs no communication, as opposed to the first few which can be very verbose and may need a great deal of interaction. We try to adopt a coordination scheme that falls between these two extremes and can adapt itself to the dynamic situation.

Our coordination scheme is based mainly on the 'coordination without communication' model of Rosenschein and Ginsberg [1988; 1987]. This approach is derived from game theory and assumes agents have a complete knowledge of each others' data. We use a slightly weaker assumption and try to arrive at complete knowledge (or at least, complete enough for our purposes) starting from incomplete or erroneous knowledge.

The foundation of 'cooperation without communication' is the computation and analysis of payoff matrices. These are representations of the benefit each agent derives from the possible combinations of moves[1] available at any point in the planning process, each agent chosing one move. Ginsberg [1987] gives a more complete explanation of payoff matrices and their use in cooperation.

4.2 Using payoff matrices

Agents use payoff matrices to decide which agent should plan which subgoal. At each choice point, agents have a choice of waypoints to plan the next bit of the path. Each choice corresponds to a distinct move by the agent. All the agents working on the plan have similar sets of choices and can guess the choices available to other agents, and can therefore compute the payoffs for each combination of choices[2] and use the results to construct a payoff matrix for that choice point. These payoff matrices help agents decide the overall best move for each agent, which may not be the same as the single best move; in multiple agent systems, it might be better for the system as a whole if some agents examine lower-worth choices rather than each agent simply choosing its most profitable move.

Our agents use three factors in calculating payoff matrices:

- The inherent cost of the move. In route planning, this is the distance between points.
- The certainty associated with the cost calculation. This allows agents to use expected utility rather than a fixed value which may be wrong.
- The fact that they are working in a multiple-agent system. This implies that agents take into consideration the expected plans of other agents when selecting a goal for expansion. The consequences of this depend on the rationality of the agents; if the aim is to distribute load so that no task is done by more than one agent, the agents pick tasks no other agent is expected to do; if the agents are working in a supportive mode, with one agent checking the other's work, they pick the same task that the other is expected to pick.

4.3 Conflict resolution using payoff matrices

We calculate payoff matrices in much the same manner as Rosenschein and Breese [1988] but use the difference

[1] At some points of the planning process, the planner may have a choice of alternatives to explore next. We call each such alternative a move available to the agent at that point. If there are n agents, the payoff matrix is a mapping from the cartesian product of the n sets of available moves to $\Re^n$.

[2] Not necessarily with complete accuracy.

between payoffs for different agents, for the same combination of moves, as a guide to decide which move or moves should be reevaluated [Malyankar, 1989]. Thus after making the matrix, the agent compares payoffs calculated for itself and for the other agent and collects cells for which the ranking is different. The goals corresponding to the cell with the most difference in ranking are then passed to the reevaluation module which takes over the responsibility of finding out why the discrepancy occurred and what can be done about it. This approach has the advantage of going into reevaluation only if the discrepancies are large enough to warrant a change in the ranking of goals. One disadvantage is that rankings are perhaps not the best things to use if we are going to use differences to decide which move should be checked for possible disagreements; actual payoffs might be a better criterion. But since we want to restrict reevaluation to moves that will repay examination, we want to avoid reevaluating useless moves already far down the list of choices for both agents.

5 Reevaluation of goals and conflict resolution

Goals are reevaluated by finding potential causes for discrepancy and collecting the relevant data from other agents before re-calculating costs. The heuristics that guide reevaluation are encoded in the form of 'reevaluation rules' like: "if there is an object O that overlaps the area around the prospective path, and the other agent has a different idea of its location from me, then set up a conversation to decide the exact location of O". It should be pointed out that the agent has not yet communicated with any other agent but uses these rules on its own world model. Applying the rules leads to identifying an agent which may be able to help in resolving the conflict and a interaction with this agent is carried out. Agents have scripts that are templates for the interactions whereby these conflicts can be resolved. These templates are used in the actual resolution interactions after values have been bound to the variables in the templates.

5.1 Strategies for conflict resolution

The strategies for dealing with conflict detection and resolution are either goal-driven, in the sense that agents try to detect and resolve conflicts depending on the goals they will work on, or conflict-driven, in the sense that agents interact to detect and resolve conflicts only when they have reason to expect a conflict. The strategies are described below.

5.1.1 Immediate resolution

This is a conflict-driven strategy. The reevaluation rules are applied as soon as conflicts are detected and transaction records made. The transaction records created by the rules are taken up and the database updated. After all transactions have been completed, costs for goals are recomputed. The main defect of this approach is the necessity to recompute costs for goals after each resolution step, which is usually computationally costly. On the other hand, the convergence to agreement of different agents is faster since the database gets updated more frequently. This decreases the chance of disagreements in the later stages.

5.1.2 Delayed resolution

This too is conflict-driven, and is a variation of the first strategy. Agents plan a few steps before any resolution is done. Conflicts detected during the planning of any single step are recorded and resolved *en masse* after an arbitrary number of cycles. The success of this strategy depends on the circumstances of the problem; it may formulate plans that are less optimal than immediate resolution allows.

5.1.3 Distributed data model

This is a goal-driven strategy. Agents evaluate their knowledge and that of other agents relative to the choices generated at every choice point. If they decide that another agent might know more about the data needed to plan any specific choice than they do, they ask that agent for information and use it to update their databases. This strategy is more 'correct' in the sense that any plan made is more likely to be optimal than under the other two strategies.

This approach suffers from the defect of having the potential for a large number of messages. On the other hand, if the task is contained more or less in a single agent's area, very few messages are sent or received, which is generally a good thing. But since agents do not keep records of data, the same data might be requested over and over again; this is offset by the tight control over updates to the authoritative version maintained by the agent responsible for it.

6 Implementation

The system is implemented in UNH Prolog on a SUN 3/50, using the SunWindows environment to run different agents in different windows. Message-passing is implemented by means of specially written built-in functions. Agents work in real time and the message-passing functions are non-blocking. Agents are differentiated from each other by loading a (different) single data file containing agent identity for the specific agent while having most other data and functions in common. Messages are logged and counted. Processing time is measured using the 'gettimeofday' system call, so that the time measured is real time.

7 Experiments and evaluation

Several factors must be considered in evaluating a planning system. For the planning component, these include the quality of plans and the time consumed by planning. For a multiple agent planner, we should also consider the communication load, since this is often a limiting factor in real systems; the granularity of processing, which affects the time agents spend waiting for replies from other agents; and the overall efficiency of the system, including the amount of duplication of work and availability of subproblem solutions.

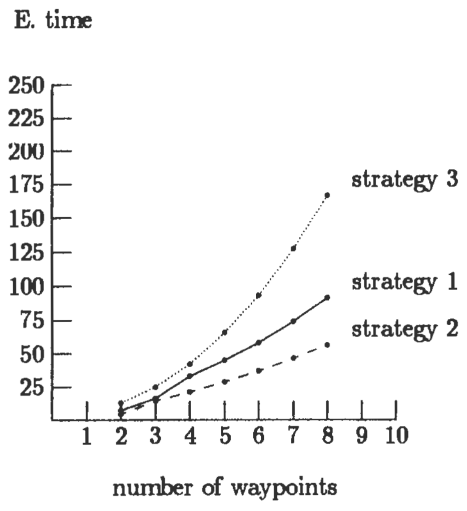


Figure 2. Waypoint-time plots, 3 obstacles

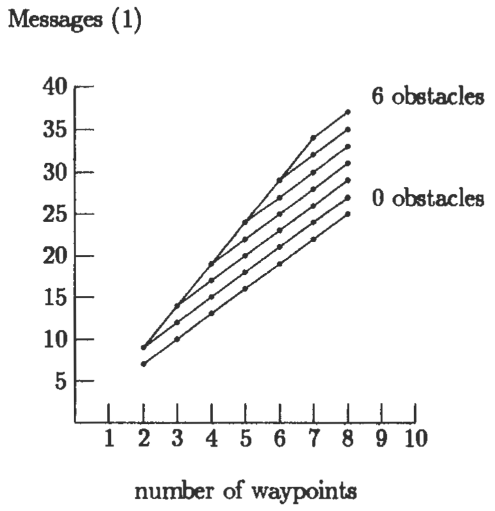


Figure 4. Waypoint-message graph, strategy 1

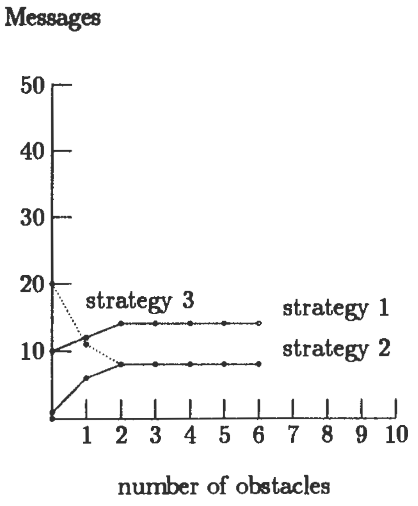


Figure 3. Obstacle-message plots, 3 nodes

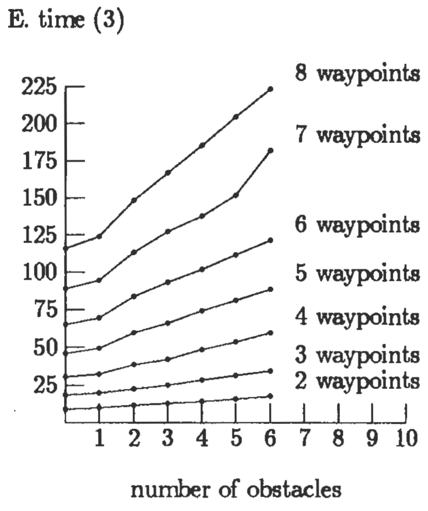


Figure 5. Obstacle-time graph, strategy 3

7.1 Experiments and Results

All three strategies (conflict-driven immediate and delayed resolution and goal-driven resolution) were run on variations of the same scene, with the number of waypoints and obstacles increasing from three waypoints and no obstacles to six waypoints and six obstacles. The waypoints are laid out so as to present the planner with two almost equally good alternatives at each choice point. The obstacles are placed so as to cause a conflict at each choice point. The full scene is described in [Malyankar, 1989]. Elapsed time was measured and the number of messages counted for various numbers of waypoints and obstacles.

7.2 Behavior with changes in problem size and number of obstacles

We plotted graphs for the variation of elapsed time and the number of messages with the problem size, as represented by the number of intermediate waypoints (those other than the start and finish), and with the number of obstacles. All these variations are presented in [Malyankar, 1989]. Some sample graphs are shown in figures 2–5. These graphs show:

- Piecewise behavior, depending on the local characteristics of the problem, namely the presence or absence of conflict and the relevance of obstacles to planning. This is apparent in Figures 3 and 4, and indicates that the strategies react to changes in problem characteristics and can to some extent allow tighter or looser interaction as required. The graphs for strategy 2 (resolution at intervals) also show discontinuities at the interval of resolution.
- A linear monotonically increasing relationship between time and significant obstacles[3] when problem size is constant; a monotonically increasing relationship between the time and problem size. This is based on the graphs in figure 2 and 5 (and the corresponding graphs for the other strategies in [Malyankar, 1989]).
- A similar relationship between the number of significant obstacles and the number of messages, except for strategy 3, where the curve is descending rather than ascending (figure 3).
- A distinct time advantage for the conflict-driven methods of cooperation (strategies (1) and (2)) over the goal-driven strategy (3), as can be seen from figure 2.
- A switch in communication load advantage between the conflict-driven and plan-driven strategies (figure 3), depending on the heuristic used in strategy (3) to decide whether communication should be done.

7.3 Timings

The most obvious result is that the goal-driven strategy usually takes much longer than any other method. This is not surprising since the agent here must make a plan as well as do the processing associated with obtaining data from other agents and integrating that data into its own database, for every goal generated at a choice point. These graphs also indicate that delayed resolution (strategy 2) is usually faster than immediate resolution (see figure 2).

7.4 Messages

The results for messages are the same as for time, for smaller problems with few obstacles. However, because of the heuristic used to guide requests for information in the goal-driven model, the number of messages actually falls with an increase in the number of obstacles, provided the size of the problem remains the same (figure 3). This is because the heuristic used is based on the existence of obstacles to guide coordination; the lack of obstacles means the heuristic cannot differentiate between agents and the default action is taken, which is a request for information from the other.

[3]Defined as obstacles which give rise to conflicts.

7.5 Quality of plans

The immediate method makes the best possible plan within the limitations of the planner. This is because information is available immediately. The delay method plans a number of steps before obtaining more up-to-date information and may lead to a deterioration in the quality of the plan. This is traded against solution time and granularity. The number of errors made depends on the layout of waypoints and obstacles and the resolution interval. The scene used allows at most two errors.

7.6 Significance of results

Since it is very difficult to define an 'average problem' in route planning with obstacles, these results cannot be taken as definitive measures of the behaviors of the strategies they correspond to for a fixed problem size. The actual behavior of the strategies will depend greatly on the actual circumstances.

What the results presented here do is define the limits of behavior of the different strategies and show the way in which that behavior varies with changes in the parameters of the problem. The graphs in this chapter are quite regular, but this regularity arises from the artificial construction of the problems on which the different strategies have been tested. It is quite possible to construct a scene that allows the strategies to exhibit one behavior while solving one part of the problem and a different behavior on a different part; the piecewise nature of many of the graphs shows how this can happen. However, since the problems span a range of possibilities—from problems without the need for any conflict resolution to those where conflict resolution is needed at every step—we consider that these sets of graph do indeed define the limits on the behavior of the strategies under different conditions. It should be possible to predict the behavior of the various strategies on more realistic non-uniform scenes from the appropriate parts of the appropriate characteristic curves.

As for the quality of plans made, this varies depending on the circumstances of the problem. The scene used for experiments shows only minor deterioration in the quality of planning for delayed resolution.

8 Relation to other research

This section discusses the relation to work in conflict resolution; work in coordination is discussed in section 4. Much of the work in distributed artificial intelligence has concentrated on frameworks for distributed problem solving [Davis and Smith, 1983; Durfee *et al.*, 1987; Durfee and Lesser, 1987; Lo and Findler, 1987; Rosenschein and Breese, 1988], on such matters as problem decomposition and distribution of subtasks. Conflict resolution has been studied by Adler and others [1988], but this has so far been qualitative, in that the aim is simply to build a framework for conflict resolution and study how conflicts can be resolved. The research described in this paper attempts to construct a more generalized framework for both cooperation and conflict resolution in the context of route planning using a combination of techniques and attempts a quantitative characterization of the behavior of the system constructed on problems in its domain.

9 Conclusion and Future Work

We have implemented and tested methods of conflict avoidance and resolution in a distributed planning system. The results indicate that given the assumptions of independent agents and conflicting data, conflict-driven resolution functions better than goal-driven resolution. Of the two conflict-driven methods tried, the rigorous method of immediate resolution makes better plans than the potentially faulty method of delaying resolution, at the expense of more time and a higher communications load.

Our system uses only two agents; it would be interesting to extend the methods to more agents and use the calculations of payoffs to distribute tasks among agents. This would free agents to take up other parallel tasks and should allow strategies that distribute computation load. The effects of scaling to more than two agents would depend on the interaction strategy adopted; if multiagent interactions are allowed one could expect different characteristic curves. Other questions that arise are the interaction of different planning models with the coordination and resolution mechanisms and the implications for distributed planning and problem–solving in domains other than route planning.

References

[Adler *et al.*, 1988] M. R. Adler, A. B. Davis, R. Weihmayer, and R. W. Forrest. Conflict resolution strategies in non-hierarchical distributed systems. In *Collected Draft Papers of the 1988 Workshop on Distributed Artificial Intelligence.* Department of Computer Science, University of Southern California, Los Angeles, California, 1988.

[Chappell, 1987] S. G. Chappell. An autonomous vehicle supervisor using the blackboard control system. Master's thesis, Department of Computer Science, University of New Hampshire, 1987.

[Davis and Smith, 1983] R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem–solving. *Artificial Intelligence*, 20(1):63–100, 1983.

[Durfee and Lesser, 1987] E. Durfee and V. R. Lesser. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 875–883. Morgan Kaufmann, San Mateo, California, 1987.

[Durfee *et al.*, 1987] E. Durfee, V. R. Lesser, and D. D. Corkill. Coherent cooperation among distributed problem–solvers. *IEEE Transactions on Computers*, 36:1275–1291, 1987.

[Ginsberg, 1987] M. L. Ginsberg. Decision procedures. In M. N. Huhns, editor, *Distributed Artificial Intelligence*, pages 3–28. Pitman Publishing, London, U. K., 1987.

[Hayes-Roth, 1985] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26:251–321, 1985.

[Lo and Findler, 1987] R. Lo and N. V. Findler. Empirical studies on distributed planning for air traffic control. Technical report, Department of Computer Science, Arizona State University, 1987. Parts 1–3, Technical Report TR–87–007.

[Malyankar, 1989] R. M. Malyankar. Cooperative route planning by multiple agents. Master's thesis, Department of Computer Science, University of New Hampshire, 1989.

[Rosenschein and Breese, 1988] J. S. Rosenschein and J. S. Breese. Communication–free interaction among rational agents: A probabilistic approach. In *Collected Draft Papers of the 1988 Workshop on Distributed Artificial Intelligence.* Department of Computer Science, University of Southern California, Los Angeles, California, 1988.

[Simmons and Chappell, 1988] A. B. Simmons and S.G. Chappell. Artificial intelligence–definitions and practice. *IEEE Journal of Oceanic Engineering*, 13:14–42, April 1988.

Solving the Problem of Hierarchical Inaccuracy in Planning*

Qiang Yang
Department of Computer Science
University of Waterloo
Waterloo, Ontario, N2L 3G1
Canada

Abstract

In *hierarchical planning* systems, a simplistic application of the STRIPS assumption can cause many computational problems. This is because changes to the truth value of a condition during hierarchical planning may only be available after certain actions are reduced. Therefore, certain control decisions based on the truth value of these conditions may have to be undone later in a planning process. To solve this problem, we present a set of syntactic restrictions on how the actions of a planner should be related via a set of action reduction schemata. When these restrictions are satisfied, a number of decisions can be made in a more informed way during planning. Furthermore, these restrictions are syntactic in nature, and they enable a set of efficient algorithms to be used for preprocessing the planning knowledge *a priori*.

1 Introduction

In most classical planning systems, the STRIPS assumption ([Fikes and Nilsson, 1971; Wilkins, 1988]) plays an important role. This assumption states that the truth value of a predicate is not changed after an action is executed, unless the change is stated explicitly in the effects of the action. It allows one to decide quickly the truth of a given condition, and is an elegant solution to the difficult frame problem.

To achieve efficiency, a planner is often implemented in a hierarchical form ([Tate, 1977; Sacerdoti, 1977; Wilkins, 1984; Charniak and McDermott, 1985]). In this form, a high-level action is associated with a few important conditions to be achieved. Planning is done in a top-down process, in which a plan is built with high-level actions first, and then refined to include more and more detailed information. The refinement is done according to a set of action reduction schemata, which defines the relationship between a high-level action and a set of low level ones. With a hierarchical planner, a simplistic application of the STRIPS assumption may no longer be sufficient; sometimes a seemingly correct conclusion about the truth of a condition may turn out to be incorrect after certain actions are reduced to lower level ones.

As an example, consider a plan for shopping involving three actions: obtaining money, getting to the shopping mall, and doing the shopping, in that order. Suppose the third action, doing the shopping, is refined before the second one, getting to the shopping mall. Then the planner will make shopping decisions based on the belief that money is available. However, after the second action is refined, the planner may then realize that getting to the shopping mall may take so much money that many of its shopping decisions have to be undone. Thus, backtracking becomes necessary.

The above problem, which will be called *hierarchical inaccuracy*, is caused by the lack of certain vital information at the higher levels of problem solving. Omission of certain information is a by-product of problem solving via abstraction. This problem solving strategy is aimed at tackling certain important parts of the problems first, before details are considered, and has been proved to be an effective means of achieving efficiency. However, if one is not careful in defining the abstraction hierarchy, serious computational problems can occur. The problem of hierarchical inaccuracy occurs when a high-level action fails to represent the change to a condition by one of its subactions, when this change is considered important during planning. Therefore, at a given planning level, a planner may make decisions based upon the currently known status of that condition. This may be a mistake, depending on how the condition is changed by the subaction of the high-level action, and backtracking may have to be used later in the process.

To solve the problem, several solutions have been proposed. One solution, given in [Wilkins, 1988], is to introduce a "delay operator", which delays reducing certain non-primitive actions until enough information is available. The problem with this method is that it is not clear how it can be generalized to domains other than the one example given in [Wilkins, 1988]. In other words, the ways to introduce this operator appear to be rather domain-dependent. Another solution proposed by Wilkins amounts to "promoting" certain conditions

*Support for this research was provided in part NSF Presidential Young Investigator award DCR-83-51463 to Dana Nau, and in part by an interim research grant from the Faculty of Mathematics at the University of Waterloo.

that may cause trouble during planning, so they can be represented higher in the hierarchy. Unfortunately, only a single example of this method was provided, and no formal elaboration made. This approach, however, is similar to that which we consider in this paper.

In [Charniak and McDermott, 1985], another solution was discussed, by setting up an "assumption protection interval." Such an interval makes sure that the expansions chosen for the non-primitive actions within its range will not undo the conditions being protected. Such a method works well in some domains, but backtrackings still occur when no good expansions are available.

Our solution to the problem of hierarchical inaccuracy is to preprocess the planning knowledge of a planner, according to a set of syntactic restrictions. These restrictions further restrict the relationship between each action and its set of subactions. They enable the problems described to be recognized early, before the planning process begins. Thus, one can anticipate the problem in a number of ways during planning.

Such a strategy is appealing for several reasons. First, it is domain-independent. Secondly, there are usually several ways of defining a hierarchical representation for a given domain of application. It may not be obvious to the designer which way is better. The restrictions provided here can serve as a standard for how to design the planning knowledge base, as well as suggestions for how the modifications can be done when some restrictions are not satisfied. Thirdly, they provide efficient control strategy during planning. In particular, they allow a planner to make better decisions about ordering between actions for the purpose of handling conflicts. They also allow intelligent decisions regarding choice of reduction schema of a non-primitive action. Finally, certain decisions have to be made about which actions to expand next in a plan, in the next planning cycle. This kind of decision can be called planning orders, since they decide which subgoal should be planned for first. Based on the restrictions, methods can be devised for deciding good planning orders.

In this paper, we present these restrictions, algorithms for preprocessing the planning knowledge based on the restrictions, as well a discussion on how the restrictions are used during planning.

2 Hierarchical Planning

As in any planning system, an important component of a hierarchical planner is a set of action templates Λ. Let $\mathbf{a}$ be an action template. Then $\mathbf{a}$ has a set of preconditions and effects, which will be denoted as preconditions($\mathbf{a}$) and effects($\mathbf{a}$), respectively. In this paper, we assume each precondition or effect is a literal. Thus, for example, one can choose to represent the action template for moving a block x from the top of another block y to the table in the blocks-world domain as

put-block-on-table(x, y)
  comment: *move x from top of y to the table.*
  preconditions=$\{\text{Block}(x), \text{Block}(y), \text{Cleartop}(x), \text{On}(x,y)\}$
  effects=$\{\text{Ontable}(x), \text{Cleartop}(y), \neg\text{On}(x,y)\}$

An action template in Λ can be *instantiated* by replacing some of the variables in its preconditions or effects by constants. An action instance is also called an *action* in this paper. To distinguish between the two, an action template will be denoted in boldface. If a is an instance of an action template $\mathbf{a}$, then the latter is called a *template* of the former, and template(a) = $\mathbf{a}$. It is clear that each action has a unique template.

Some of the effects of an action are the main effects, or the "purpose," of that action. If a is an action, then the main effects of a are denoted by main-effects(a). The other effects of a are called "side-effects." For example, among the effects of the action put-block-on-table(x, y), Ontable(x) is its main effect, and the others are side-effects.

Another important component of a hierarchical planner is its definition of the relationship between its actions. This relationship is defined in terms of a set of *action reduction schemata* Φ. A reduction schema $R \in \Phi$ is a function which, when applied to an action template in Λ, returns a partially ordered set of actions a_i. Note that R is not necessarily applicable to every action template in Λ, and an action template can have more than one reduction schema applicable to it. The set of action reduction schemata applicable to $\mathbf{a}$ is denoted by $\alpha(\mathbf{a})$. $\mathbf{a}$ is *primitive* if $\alpha(\mathbf{a}) = \emptyset$, otherwise it is *non-primitive*. Intuitively, a primitive action template is one which cannot be decomposed further into more detailed steps, while a non-primitive one can.

If a is an instance of $\mathbf{a}$, and R is applicable to $\mathbf{a}$, then $R(a)$ is defined as $R(\mathbf{a})$ with the same variable instantiations as when a is obtained from $\mathbf{a}$. Also, if $R(a)$ is a reduction of a, then $A(R(a))$ denotes the set of actions in $R(a)$. These actions are called *subactions* of a.

A plan P is a partially ordered set of actions. Let a and b be two actions in a plan. Then $a \prec b$ denotes that action a is constrained to be executed before b in time. $a \preceq b$ if either $a \prec b$, or a and b are unordered in the plan.

Let a be a non-primitive action and $R(a)$ be a reduction of a. Since $R(a)$ is partially ordered, it may have more than one possible linearization. At the end of every linearization L, a set of conditions hold, which are asserted by the actions in L. We use possible-effects($R(a)$) to denote the *union* of these sets of conditions, for all the linearizations. More formally, possible-effects($R(a)$) is defined as: $\{p \mid \exists a_i \in A(R(a))$ such that$p \in$ effects(a_i), *and*$\forall a_j \in A(R(a))$ such that $a_i \prec a_j, \neg p \notin$ effects$(a_j)\}$.

The definition of reduction schemata above is intended to capture the formal aspects of action or goal expansions in a number of systems. In particular, a reduction schema R corresponds to a "soup code" in NOAH ([Sacerdoti, 1977]), an "opschema" or an "actschema" in NONLIN ([Tate, 1977]), and a "plot" in SIPE ([Wilkins, 1984; Wilkins, 1988]). Also, we make no distinction between a high-level goal and an action in our formalization; all the goals are represented as action templates in Λ which, among others, can be reduced to a special action **no-op**, a primitive action which means the action can be achieved by doing nothing.

A hierarchical planner starts planning for a given set of goals by finding the appropriate actions to achieve them. Problem solving during planning can be considered as a search in a space of plans, and is usually a very complicated process. To capture the essence, we present a brief description below. Suppose $G(X)$ is a goal, then the action selected to achieve this goal should have $G(x)$ as one of its main effects. These are the actions of a plan at the highest level. The subsequent problem solving process is described by the following steps:

1. Choose a non-primitive action, and replace it by one of its reductions. Let the new plan be P.
2. Find out the set of interactions among the actions in P, and find ways to handle them.
3. If all the actions in P are primitive then terminate planning, else go to step 1.

After step 1 is done, certain new interactions may appear. Depending on the particular domain of application, interactions can be of different types. In this paper, the only type of interaction considered is clobbering between a pair of actions. This type of interaction occurs when an action in a plan deletes a precondition or a main effect of some other action. To take care of a conflict, several methods can be chosen, including imposing new orderings in the plan, imposing constraints on the variable instantiations in the plan, etc.

Below, we will consider how to impose syntactic restrictions upon the planning hierarchies that define clearly the relationship between a non-primitive action and its set of subactions. Then a set of algorithms will be given for checking whether a reduction hierarchy satisfies the given restrictions.

3 Imposing Syntactic Restrictions

In hierarchical planning, a currently non-interacting part of a plan may become interacting after certain actions are reduced. This problem of hierarchical inaccuracy can then causes problem in several crucial decision processes during planning. We propose to handle this problem by checking pairs of action templates for a given set of action reduction schemata definitions. If the problem of hierarchical inaccuracy cannot occur between this action pair, then they are marked as so. In this section, we discuss the restrictions in detail, then we consider how to apply the result of the preprocessing phase to help planning.

Let P be a plan containing certain non-primitive actions. As is done in all existing hierarchical planners, we assume that when an ordering $a \prec b$ is assigned in P, all the subactions of action a will precede that of action b. Therefore, at any later stage of plan reduction, the effects of any of the subactions of b cannot delete the preconditions of the subactions of a, and the effects of the subactions of a cannot delete any of the effects of the subactions of b. However, it is possible that some subactions of a may delete some preconditions of b, and certain subactions of b may delete the effects of the action a which are protected. Thus, it is important to know when an ordering can be "safely" made.

Now we state this property more formally.

Definition 3.1 *Let P be a plan, and $a, b \in A(P)$ actions in P. The* Ordering-Induced Independence Property holds for (a,b) iff the following conditions are satisfied: for any composite reductions Q_1 and Q_2 of a and b, respectively,

1. $\forall p \in$ preconditions$(b), \forall q \in$ possible-effects$(Q_1(a))$, $\neg p$ and q are not unifiable, and
2. $\forall q \in$ main-effects(a), $\neg q$ is not unifiable with any $p \in$ possible-effects$(Q_2(b))$.

This definition says that if (a,b) satisfies the Ordering-Induced Independence Property, and $a \prec b$ in a plan, then no subactions of a can conflict with any preconditions of b, and no subactions of b can conflict with any main effects of a, which has to persist over the reduction of b. Therefore, if p is an effect of a, and $a \prec b$ in a plan, then it is safe to assume that p is also true after any reductions of b. Notice that since p and q are literals, checking whether or not they are unifiable is straightforward.

To ensure the Ordering-Induced Independence Property for two actions, we introduce two restrictions below:

Restriction 3.2 *Let a and b be two actions. $\forall R \in \alpha(a)$, $\forall p \in$ possible-effects$(R(a))$, and $\forall q \in$ preconditions(b), if p and $\neg q$ are unifiable, then $p \in$ effects(a).*

This restriction says that if any subaction a_i of a can possibly deny a precondition of b, then this effect of a_i should also be an effect of a. If this restriction is satisfied by actions a and b, then we say that a satisfies Restriction 3.2 with respect to b. Note that it is possible that p is an effect of a_i and unifiable with q, but p is not in the set possible-effects$(R(a))$. This can occur when there is another subaction a_j of b such that $a_i \prec a_j$ and $q \in$ effects(a_j). In this case, we do not require that p is also in effects(a).

Restriction 3.3 *Let a and b be two actions. $\forall R \in \alpha(b)$, $\forall q \in$ effects$(R(b))$, and $\forall p \in$ main-effects(a), if p and $\neg q$ are unifiable, $q \in$ effects(b).*

This restriction says that if a subaction b_j of b can possibly deny a main effect of a, then this effect of b_j should also be an effect of b. If a and b satisfy this restriction, then we say that a satisfies Restriction 3.3 with respect to b. If actions a and b satisfy both of these restrictions, then we say (a,b) satisfies the *Condition-Promotion Restriction.*

The Condition-Promotion Restriction further restricts the relationship between an action and its set of subactions in adjacent levels. Below, we prove that this ensures the Ordering-Induced Independence Property over arbitrary levels of reduction. Before doing this, we first introduce a new notation TC(a), representing the *transitive closure* of the action-subaction relationship.

Now we present the main theorem:

Theorem 3.4
The Ordering-Induced Independence Property holds for (a,b) if $\forall \mathbf{a_i} \in$ TC(template(a)), $\forall \mathbf{b_j} \in$ TC(template(b)), $(\mathbf{a_i}, \mathbf{b_j})$ satisfies the Condition-Promotion Restriction.

The proof of the theorem is omitted. Interested readers can refer to [Yang, 1990].

There are several ways to make use of the results of the above theorem. First, orderings have to be frequently assigned in a plan, and Ordering-Induced Independence Property provides a good heuristic for making the ordering decision when no other preferences are known. For example, if a and b are unordered in a plan, and they satisfy the Ordering-Induced Independence Property, then if a choice for ordering have to be made involving a and b, this choice will be preferred.

As an example, consider a plan for achieving two goals, writing a letter and developing a film. An intermediate hierarchical plan may contain two parallel branches, where the first branch is:

$$\text{get(Pen)} \prec \text{write(Letter)},$$

and the second branch can be:

$$\text{get(Tools)} \prec \text{develop(Film)}.$$

If the robot can hold only one thing at a time, then these two branches cannot be interleaved. Thus, a choice has to be made in ordering: either write(Letter) has to come before get(Tools), or develop(Film) has to come before get(Pen). Suppose a subaction of develop(Film) is to turn off the light, which asserts a condition $\neg$On(Light), and that this condition is not represented in the effects of develop(Film). Also suppose that a precondition of get(Pen) is that the light must be on. Then the action pair (develop(Film), get(Pen)) does not satisfy the Condition-Promotion Restriction. On the other hand, suppose (write(Letter), get(Tools)) satisfies the Ordering-Induced Independence restriction. Then a good choice is to order write(Letter) before get(Tools). In this way, a potential conflict is avoided.

If ordering decisions are made correctly, then it is possible for a planner to avoid an exponential amount of conflict checks done in a partially ordered plan. To illustrate this, consider a hierarchical planning problem with k levels of reduction, and suppose that each non-primitive action can be reduced into m subactions at the level below. Suppose the levels of reduction are numbered in ascending order from top-down, so that the bottom level is Level k and the top level is Level 0. Suppose a hierarchical nonlinear planner finishes removing all the goal interactions on the i'th level before it goes into the $(i+1)^{st}$ level. An action at the i'th level will be reduced into m^i actions at Level k. Therefore, each time-ordering arc which can be introduced at the i'th level avoids $(m^{k-i})^2$ number of interaction checks at the k'th level. Thus, if the planner can add e time ordering arcs at each level of abstraction, the total amount of planning time saved will be

$$\sum_{i=0}^{k-1}(em^{2(k-i)}) = O(em^{2k}).$$

Secondly, decisions as to whether a condition holds at a certain point in a plan have to be made during planning. The Ordering-Induced Independence Property allows such decisions to be made in many cases, even before every action in a plan is reduced to primitive. Suppose an action a asserts a condition p in a plan, and one wishes to decide whether p holds at a later point, say before action c, in the same plan. If for every action b that can be possibly between a and c, (a, b) satisfy the Ordering-Induced Independence Property, then if p is not deleted by b in the current plan, then p will not be deleted by any subactions of b either. This argument leads to the following lemma:

Lemma 3.5 *Let P be a plan, a and c be actions in the plan. Suppose that $a \prec c$ and $p \in$* main-effects(a). *Also suppose that $\forall b \in A(P)$, $a \preceq b$ and $b \preceq c$, (a, b) satisfies the* Ordering-Induced Independence Property. Then if p is true before c in plan P, then p is true before c in every composite reduction of P.

One necessary decision to be made regarding the truth of a condition is that of which reduction R of an action a should be chosen, among the set of alternative reductions of a, $\alpha(a)$. Usually a reduction is associated with a type of condition called "use-when", which tells when the reduction is applicable to a non-primitive action. When the truth value of a use-when condition is decided incorrectly, or cannot be decided at a certain level of planning, reductions have to be arbitrarily chosen. This can lead to many expensive backtrackings. The Ordering-Induced Independence Property, where it holds, can help solve part of this problem by using the above lemma.

Another type of decision is when there are more than one non-primitive actions to be reduced, which one should be reduced first at a certain planning level. Decisions like this are called "planning order," as opposed to the temporal orders in a plan. Suppose that $a \prec b \prec c$ in a plan, and both b and c are non-primitive actions. Suppose also that an effect of a is p, and b does not deny this condition. Then according to the STRIPS assumption, p can be assumed to hold before c on this planning level. Suppose further that a and b satisfy the Ordering-Induced Independence Property, and a reduction of c depends on the truth of p. Then that reduction can be chosen, because reducing b further will keep the validity of p.

If, on the other hand, a and b do not satisfy the Ordering-Induced Independence Property, then it is suggested that the decision about reducing c should be deferred until b is further reduced. An appropriate time to reduce c may be when for every action b_i between a and c, a and b_i satisfy the Ordering-Induced Independence Property. This avoids many possible backtrackings, due to the decisions made with incomplete information in the original planning level.

Consider the following simplified shopping example. In this simplified domain, a robot wants to shop for food. There is a shop, Shop, which the robot can reach by a bus or a bike. Initially the robot has money. The actions and reduction schemata are listed in Appendix A. Note that the action pair (goto(y), buy(x)) does not satisfy the Condition-Promotion Restriction, because a possible descendent take-bus(y) of goto(y) can delete a precondition of buy(x), namely, Have(Money), while this change is not seen in the effects of goto(y). Thus, it is not known whether these two actions satisfy the Ordering-Induced Independence Property.

During planning, an intermediate plan for shopping for food is given below:

$$\text{goto(Shop)} \prec \text{buy}(x).$$

At this stage of planning, since it is *not* known that the action pair satisfies the Ordering-Induced Independence Property, the planner cannot conclude that Have(Money) is still true before buy(x) is executed. Thus, for example, the planner may want to try to get more information about goto(Shop) by reducing it first, before reducing buy(x) to make detailed shopping decisions.

A remaining question is how to preprocess a set of action reduction schemata based on the Condition-Promotion Restriction. In the following sections, we provide algorithms for doing this.

4 Preprocessing

We have shown that if all the information regarding whether the actions satisfy the Ordering-Induced Independence Property is gathered before planning starts, then a planner can make decisions more intelligently during planning, based on this information. Specifically, if the Condition-Promotion Restriction is satisfied by a and b, then the ordering $a \prec b$ will never be undone as a result of the interactions between the subactions of a and/or b.

Let $\mathbf{a}$ and $\mathbf{b}$ be two action templates in Λ. Suppose the number of effects and preconditions of an action is constant. Also suppose that the maximum number of actions in a reduction is H. The following algorithm checks if $(\mathbf{a}, \mathbf{b})$ satisfy the Condition-Promotion Restriction:

Algorithm Checking

1. Compute TC($\mathbf{a}$) and TC($\mathbf{b}$). This step can be done in time $O(k \times H \times |\Lambda|)$ using depth first search, where k is the maximum number of reduction schemata applicable to an action, and H is the maximum number of subactions in the reduction of a non-primitive action.
2. Check for every action template $\mathbf{a_i} \in$ TC($\mathbf{a}$), $(\mathbf{a_i}, \mathbf{b})$ satisfies Restriction 3.2 with respect to $\mathbf{b}$. To check the restriction for each pair $(\mathbf{a_i}, \mathbf{b})$, one can first compute

   $$E = (\bigcup_{R \in \alpha(\mathbf{a_i})} \text{possible-effects}(R(\mathbf{a_i})) - \text{effects}(\mathbf{a_i})).$$

   $(\mathbf{a_i}, \mathbf{b})$ satisfies Restriction 3.2 if $\forall p \in E$ and $\forall q \in$ preconditions($\mathbf{b}$), p and $\neg q$ are not unifiable. Notice that TC($\mathbf{a}$) can have at most $|\Lambda|$ elements. Note also that to compute possible-effects($R(\mathbf{a_i})$) requires H^3 in time complexity. Thus, for all the actions in TC($\mathbf{a}$), the total time complexity for this step is $O(k \times H^3 \times |\Lambda|)$.
3. Check if for every action template $\mathbf{b_j} \in$ TC($\mathbf{b}$), $(\mathbf{a}, \mathbf{b_j})$ satisfies Restriction 3.3. This step is similar to the previous one, and has a time complexity of $O(k \times H^3 \times |\Lambda|)$.

The total time complexity for running this algorithm is $O(k \times H^3 \times |\Lambda|)$.

If both the second and third steps of Algorithm Checking succeed, then the Ordering-Induced Independence Property holds for the pair $(\mathbf{a}, \mathbf{b})$. In that case, the pair is marked as so. Thus every pair of actions which are instances of $\mathbf{a}$ and $\mathbf{b}$ also satisfy the property.

As an example, consider the actions (goto(y), buy(x)) in Appendix A. TC(goto(y)) = {ride-bike, take-bus}. Note that ¬Have(Money) is an element of

$$\text{possible-effects}(R_2(\text{goto}(y))) - \text{effects}(\text{goto}(y)),$$

while Have(Money) ∈ preconditions(buy(x)), therefore, (goto(y), buy(x)) does not satisfy Condition-Promotion Restriction.

5 A Special Case Of Schemata Definition

A special case of action reduction schemata definition exists, which allows one to simplify the previous checking algorithm. This special case occurs when all the action templates in Λ have only positive literals as their preconditions. Then we can impose the following restriction:

Restriction 5.1 *A non-primitive action a satisfies the* Negation-Promotion *Restriction if* $\forall R \in \alpha(a)$ *and* $\forall l \in$ possible-effects($R(a)$) *such that l is a negative literal,* $l \in$ effects(a).

The result is given in the following corollary to theorem 3.4

Corollary 5.2 *Let Φ be the set of action reduction schemata, and Λ be the set of action reduction schemata for a planning system. Suppose for every action template* $\mathbf{a}$ *in Λ, all the preconditions of* $\mathbf{a}$ *are positive literals. If all the actions in Λ satisfy the Negation-Promotion Restriction, then every pair of actions in the domain satisfies the* Ordering-Induced Independence Property.

The advantage of this corollary is that it allows a set of action reduction schemata to be preprocessed in a more efficient manner. In particular, the Negation-Promotion Restriction only restricts the way each action is related to its own set of subactions in its reduction schema. For each element p of possible-effects($R(a)$), if p is a negative literal, then a check can be made to see if p is also in effects(a). If every action in TC(a) satisfies the Negation-Promotion Restriction, then a is marked as so. If a and b are both marked, then both (a, b) and (b, a) satisfy the Ordering-Induced Independence Property. To examine the Ordering-Induced Independence Property for every pair of action templates, this marking algorithm need only run $|\Lambda|$ times, while in the previous section, the algorithm Checking has to run $|\Lambda|^2$ times in order to check every pair of actions. Note that the action goto(x) in the shopping example does not satisfy the Negation-Promotion Restriction.

6 Conclusion

This paper presented a solution for the problem of hierarchical inaccuracy in hierarchical planning, which occurs when changes to a critical condition are not appropriately represented at the upper levels of knowledge representation. Allowing this to occur may cause many backtrackings during planning, and thus may detract from planning efficiency.

Our solution to this problem is to impose syntactic restrictions on the definition of the relationship between a high-level action and its reductions. These restrictions are used to check if some subactions of a non-primitive action can affect a condition in an unexpected way during planning. An important feature of these restrictions is that they are syntactic, thus enable us to incorporate them in a preprocess algorithm. We also demonstrate how the preprocessed planning knowledge can be used in a number of ways for more informed decision making.

The results presented in the paper can be useful for automatically constructing planning hierarchies, given a set of primitive actions as input. Usually for a set of actions, there may be several ways for building this hierarchy, and it is not obvious which alternative is better. For example, it may not be clear whether a particular effect of an action should be considered important, and should be promoted to a higher level. Condition-Promotion Restriction can be used to check which alternative hierarchy is superior; if one way of building the hierarchy satisfies the restriction, then it may be preferred. We will pursue this issue further in our future research.

Acknowledgements

The author is very grateful for the many suggestions given by Dana Nau and James Hendler. Many thanks also to Jill Yang for her careful editing.

References

[Charniak and McDermott, 1985] Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence.* Addison-Wesley Publishing Company, 1985.

[Fikes and Nilsson, 1971] Richard Fikes and Nils Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Sacerdoti, 1977] Earl Sacerdoti. *A Structure for Plans and Behavior.* American Elsevier, 1977.

[Tate, 1977] Austin Tate. Generating project networks. In *Proceedings of the 5th IJCAI*, pages 888–893, 1977.

[Wilkins, 1984] David Wilkins. Domain-independent planning: Representation and plan generation. *Artificial Intelligence*, 22, 1984.

[Wilkins, 1988] David Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm.* Morgan Kaufmann, CA, 1988.

[Yang, 1990] Qiang Yang. A solution to the problem of hierarchical inaccuracy in planning. *in preparation*, 1990.

A Action Templates and Their Reduction Schemata for a Shopping Domain

Here we define part of a simplified shopping domain, where there is a shop, Shop, and one can go to the shop either by bus or by riding a bike. Below we list the non-primitive and primitive actions, along with a set of action reduction schemata. If an effect is a main effect of an action, then it is marked by "*".

The following actions are non-primitive:

1. **goto(y)**
   comment: goto a place y.
   preconditions={}
   effects={At(y)}

2. **buy(x)**
   comment: buy merchandise x.
   preconditions={At(shop), Have(Money)}
   effects={*Own(x), ¬ Have(Money)}

The following actions are primitive:

1. **take-bus(y)**
   comment: take bus to place y.
   preconditions={Have(Money)}
   effects={¬Have(Money), *At(y)}

2. **ride-bike(y)**
   comment: ride bike to place y.
   preconditions={}
   effects={At(y)}

Below are reduction schemata for the non-primitive actions. A protection interval in these definitions is a condition to be protected from the end of one subaction to the beginning of another. Also, we omit the reduction schemata for the non-primitive action "buy(x)".

1. R_2**(goto(y))**
   actions: {take-bus(y)}
   orderings: {}
   protection intervals: {}

2. R_3**(goto(y))**
   actions: {ride-bike(y)}
   orderings: {}
   protection intervals: {}

A Support-System for Knowledge Engineering by the Student

Gilbert Paquette, Renaud Nadeau and Martin Longpré

Télé-université

4835 Christophe Colomb,

Montréal, Québec H2J 4C2

Abstract

We have built, and tested with students, discovery learning environments in which intelligent tools help them explore a knowledge domain and construct a rule-based system. The learner first consults a complete rule-based system. His goal is to define a correct and complete set of rules to achieve a task, without seeing the rules known to the system.

An implementation of these concepts in the PRISME programming language, and its integration in the LOUTI design system will be sketched showing the generality and usefulness of the approach. Finally, we will sketch perspectives for future applications.

1. Introduction: the student as a knowledge engineer.

In many educational applications, the knowledge that must be acquired by the learner is a rule system. One can use different pedagogic strategies to achieve that goal: by instruction, stating the rules and asking the learner to memorize; by analogy, presenting a similar system and having knowledge transferred; by induction, embedding rules in a laboratory-like environment where they can be used and re-discovered [Michalsky et al. 1983].

Because mere interaction of a learner with a knowledge-base systems (KBS), or display of its rules, are unsatisfactory ways of acquiring its knowledge, early attempts have been made to build Intelligent Tutoring Systems [Sleeman and Brown 1982; Clancey 1988]. One important unsolved problem is what John Self [1990] has labeled "the intractable student problem", that is to have the ITS maintain a sufficient student model to guide its interventions. By-passing the very need for such models, many AI applications to education have put the student in command of a discovery learning environment [Papert 1980; Borning 1981; Ennals 1983; Lawler 1987; Gaines 1988].

Although we believe exploratory environments are exciting educational tools, our previous classroom experiments has shown the importance to go beyond exploration and have the student achieve a constructive learning process [Bordier and Paquette 1989].

Our research team has spent the last three years both experimenting the use of generic software as learning tools in the classroom and building new tools using AI concepts and methods [Paquette 1988]. A design system called LOUTI has resulted from this work [Bergeron and Paquette 1989]. It is essentially a set of high level tools, built on top of the PRISME programming system [Bergeron et al. 1988], that can be assembled to create environments where knowledge bases are explored and constructed by an active learner.

Acting as a knowledge engineer in a limited domain, the student will discover important variables and relations and use them to build and validate his own rule system. Because this is a difficult task, access to knowledge, processing tools and methodological meta-knowledge must be present in the computer environment supporting the learner.

We advocate that this approach encourages the learner's motivation, attention, integration with previous knowledge and long-term retention.

This research have been achieved with the support of the APO-Québec research center and Télé-université.

2. A sample environment in physiotherapy

Based on these principles, a learning environment has been built for college physiotherapy student training. Using an ultrasonic beam machine, with the patient's variables in mind-- treatment's goal, illness status, region's depth, bone reflection phenomena - the technician must settle the five treatment variables: frequency, intensity and mode of the ultrasonic beam, time-table and duration of each application.

2.1 The learner's knowledge base

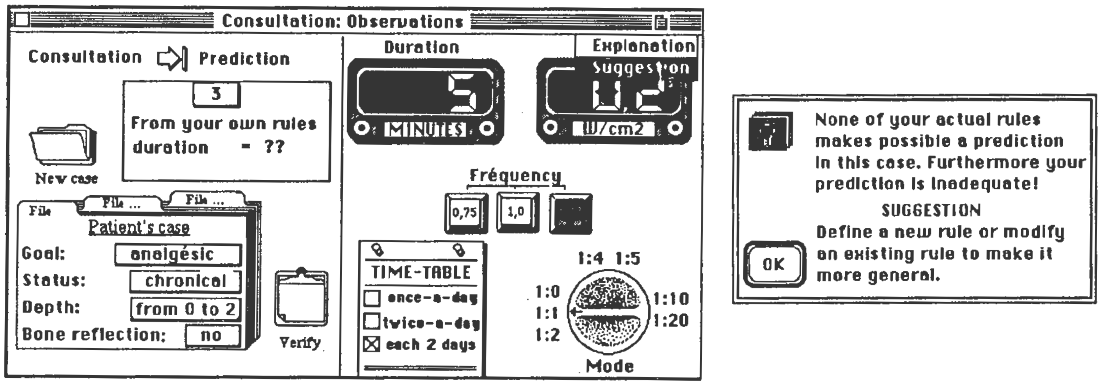
The learner starts with a set of observations chosen by the designer. Each object of this Observations's class has nine attributes, four from the patients file and the five treatment parameters.
Typically, the knowledge to be acquired is a set of if-then rules of a very simple form:

- "if the wound's depth is between 2.0 cm and 4.5 cm, then the frequency is equal to 1.0 Mh",
- "if the goal is to cure the illness and the state of the patient is chronical, then the duration of each application must be 5 minutes".

The learner starts with only these two rules as examples and he has to build a consistent and complete rule-group for each of the treatment parameters. Such groups, also called rule subjects, are defined by their treatment variable and the patient variables it depends on.
At any time, a current rule subject and a current rule is selected. These selections can be changed by the student, every tool available adapting to the new selection. The current status of the evolving knowledge base can always be viewed with presentation tools such as tables, networks or lists.

2.2 Consulting and prediction making

Right from the beginning, the system has a complete Prolog-like set of rules. They cannot be seen by the learner, but the results of the inferences based on the system's rules are accessible in a way similar to expert system consultation. A machine-like interface offers two modes: consultation and prediction. In consultation mode, the student simply settles one or more patient variables. As soon as the system can infer one or more treatment variables, he will immediately show them on the machine interface. In this way, the student can quickly discover which patient's variable(s) have influence on each parameter of the treatment and use these informations in his rule construction process.
In prediction mode, the student settles all of the patient's parameters and makes a prediction on the value of the currently selected subject. When he asks for verification, the system makes its deductions and settles the five treatment parameters on the machine. It also shows the current subject's value that should have been deduced by the student's from his own rule set.
If there are none or more than one such values, the system signals a problem by questions marks.



Consultation: Observations

3
From your own rules
duration = ??

Patient's case
Goal: analgésic
Status: chronical
Depth: from 0 to 2
Bone reflection: no

TIME-TABLE
☐ once-a-day
☐ twice-a-day
☒ each 2 days

None of your actual rules makes possible a prediction in this case. Furthermore your prediction is inadequate!
SUGGESTION
Define a new rule or modify an existing rule to make it more general.

Figure 1 - Consultation and prediction interface

The student can then ask for advice. In the example of the above figure, he has received a suggestion to use the completeness tool because the chosen case is not covered with any yet defined student rule.

2.3 Simplifying links and defining rules

Consultation mode normally leads to link identification between variables. The next figure shows a graph displayed by a typical learner at the beginning and at the end of a link simplification using the link modification tool.

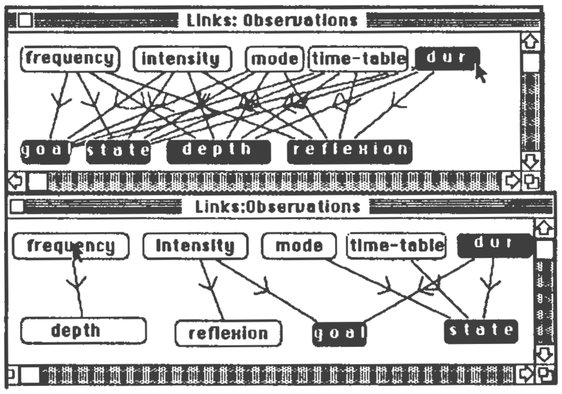


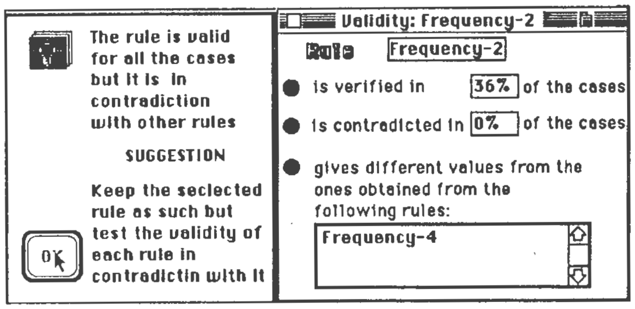
Figure 2 - Link simplification.

Simplifying dependency links in this way has many advantages. In particular, it makes the rule definition and redefinition process more focussed. Building a new rule on duration with simplified variable links reduces the choices available, and error possibilities, for the rule's attributes.

2.4 Validity and Completeness Tests

The prediction mode will normally lead the student to validity or completeness tests. If a student suspects one of his rules to fail some observations or contradict another rule, he can ask for a validity diagnosis. From the validity window, he can ask for a table of examples and/or counter-examples. He can also ask for advice from the system.

In the following example, the system says his rule is free from counter-example but is contradicted by another rule, so he should check the validity of the latter and modify or destroy it if necessary.



The rule is valid for all the cases but it is in contradiction with other rules

SUGGESTION

Keep the seclected rule as such but test the validity of each rule in contradictin with it

OK

Validity: Frequency-2

Rule Frequency-2

● is verified in 36% of the cases

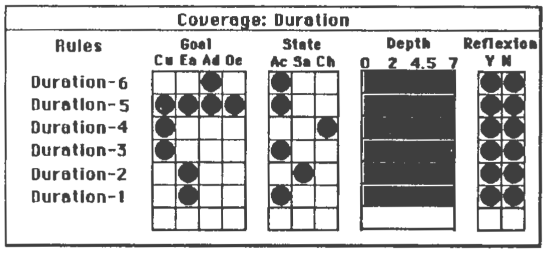
● is contradicted in 0% of the cases

● gives different values from the ones obtained from the following rules:

Frequency-4

Figure 3 - Validity test interface

The next figure shows a completeness test for the duration subject. A grid gives the attributes covered by each rule in the currently selected rule-group. In this case, one can see that many cases are covered by none of the actual rules. For example the only rule covering the case "chronical" (Ch) is rule "duration-4". In that case, when the goal is anything except "cure" (Cu), no rule is covering.

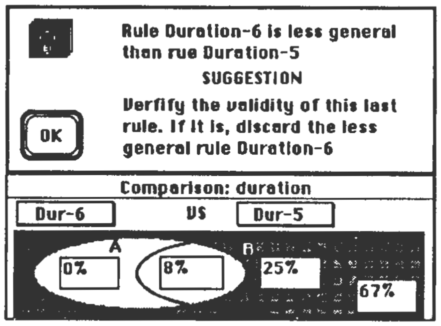


Coverage: Duration

Rules	Goal				State			Depth	Reflexion	
	Cu	Ea	Ad	Oe	Ac	Sa	Ch	0 2 4.5 7	Y	N
Duration-6			●		●			■	●	●
Duration-5	●	●	●	●	●			■	●	●
Duration-4	●						●	■	●	●
Duration-3	●				●			■	●	●
Duration-2		●				●		■	●	●
Duration-1		●			●			■	●	●

Figure 4 - Completeness test interface

On the other hand, many rules are useless. For example, if a comparison is asked between rule duration-6 and duration-5, a Venn diagram will appear showing the first to be a particular case of the second. If asked for advice, the system will tell to eliminate all these useless rules so a clearer coverage picture can be obtained on the grid.


Rule Duration-6 is less general than rue Duration-5

SUGGESTION

Verfify the validity of this last rule. If it is, discard the less general rule Duration-6

Figure 5 - Rule comparison

At any time some of the student rules can be partially or totally false or there may exist patient's cases not yet covered by any rule. The student is guided to discover such problems in many ways. First, using the rules he has already built, he can make predictions for different cases and see what happens. The system has enough knowledge to tell him if he applies his rules correctly and if the results coincide with those deduced by the system's own rules. He can also make suggestions if two or more student rule are not consistent or if none exist for to the case chosen by the student. This in turn can lead to validity and consistency test on the selected rule, to completeness check on all the rules in the selected rule-group (the selected subject) or simply a new look at the observations, especially at the rule's examples or counter-examples. One of these processes will in turn induce the student to make changes to links defining rule-schema and to modify one or more rule, define new rules or destroy old ones.

3. Integration in the LOUTI design system

The present work has resulted in an extension of the knowledge-base concept in LOUTI. We added rule-groups and rules, giving access to rule definition and processing by the learner in an integrate way with facts and concepts processing already present.

3.1 Knowledge Bases in LOUTI

The LOUTI design system gives the user (learner or designer) the possibility to build knowledge bases. The user can:

- define one or more classes grouping entities;
- add, delete or modify the class attributes;
- add elements to any class;
- define class structures: sets, relations, rule-groups and rules.

This knowledge base concept is implemented as PRISME objects.The main objects are:

- *Knowledge-base:* maintains the list of classes, sets, relations, rule-groups and rules defined in the KB; holds a currently selected type and relation; possesses functions to create, destroy or modify the KB's components.
- *Classes:* maintains a class functions (attributes) list and its elements list; holds a selected set, a selected rule-group, a list of independent and dependent variables (attributes); possesses functions to create, modify and destroy attributes and sets and to add elements.
- *Sets:* holds its class name, its definition predicate, its element lists and the relations defined on it.
- *Relations:* holds the the two sets (not necessarily of the same class) on which the binary relation is defined, its definition predicate, and the list of its couples.
- *Rule-groups:* maintains a list of its antecedent and consequent attributes, the class it is defined on, its rules list, the class elements covered by one of its rule and the current selected-rule; possesses functions to create, destroy or modify one of its rules; a rule-group can be destroy only if its rule list is empty. It can be modified by changing the antecedent and consequent lists, only if these list are supersets of all the attributes actually used in the rule-group's rules.
- *Rules:* holds its rule-group, its antecedent(if) and consequent(then) predicates; possesses functions to compute the class elements it covers and the elements on which it contradicts other rules; constructs the list of its examples, counter-examples and non-contradictions over the currently selected set of observations.

3.2 Tool management

Another very important object is "tool-management". It is responsible to alert every active tool of any modification of the knowledge base by the student. These active tools are chosen by the designer in a library which is also a hierarchy of PRISME objects.
Each library tool is linked to one of the knowledge base components. For example, a class can be seen in a set-subset tree; each set can be presented in table or graphic form, binary relations can be viewed as networks; rules can be shown in a list or evaluated using a validity rule; rule-groups can be shown on a completeness grid and their rules compared with a comparison tool.

4. Rule attribute computation

Some of the rule attributes, for instance examples and counter-examples, are computed by simply matching of the predicates against the observation set, giving the student another view of his knowledge base. But unless the observations are very well distributed, which is not always feasible, they do not give reliable induction indicators on which the system could base its suggestions to the student.
For this reason, we have developed algorithms to compute universal examples and counter-examples, coverage of a rule or divergence between rules, based on the student's rule syntax instead of the observation set. A first algorithm constructs a solution set for a predicate in a compact interval notation. A second algorithm computes a measure of the solution set from this interval notation.
But first, we will give a precise definition of some of the rule attributes we need to compute.

4.1 Definition of some rule attributes

A *rule* is an expression of the form

r(x): if p(x) then q(x)

where p(x) and q(x) are predicates constructed in the usual way from a sublist of the rule's class, PRISME relations and functions, and logical connectives.

The Cartesian product of value sets for attributes appearing in the rule is called the set of *cases*.
Each rule defines on this set of cases C three *universal characteristic sets*:

$CX(r) = P \setminus Q$, the set of counter-examples in C

$EX(r) = P \cap Q$ he set of examples in C

$NC(r) = C \setminus P$ the set of non-contradictions in C

- Note that P and Q are the solution-sets on C of predicates p and q respectively.

The *coverage of a rule* r is the subset of C for whose elements the condition predicate p holds.

$$Cov(r) = P = EX(r) \cup CX(r)$$

The *coverage of a rule-group* is the union of the coverage of each of its rules.The rule-group is said to be *complete* if and only if its coverage is equal to the rule's cases.
Last, the *divergence* between rules r_1 and r_2 of the same rule group is the set of cases for which these rules have their condition predicate satisfied, but not their conclusion predicates together.

$$Div(r_1,r_2) = P_1 \cap P_2 \cap [C \setminus (Q_1 \cap Q_2)]$$

Two rules of the same group are *consistent* if and only if their divergence is empty and a rule-group is called *consistent* if and only if any two of its rules are consistent together.

4.2 The interval form algorithm

The *interval form algorithm* takes a predicate as input and produces its solution-set in a compact interval form.

a) First, each attribute appearing in the predicate is assigned a "universal" interval of values in the following way:
• if it has bounded continuous values, the interval is defined by the lower and upper bounds;
• if it has discrete values, they are enumerated in an ordered list and ranked to produce an interval.
b) Next, depending on the attributes and relations it uses, each atomic predicate is assigned a list of sub-intervals for each attribute that represent the cartesian product of the sub-intervals.
c) Finally, the logical connectives linking the atomic predicates are taken in account, and simplification rules are used, to produce a three level list of intervals representing the solution-set of the input predicate.

Applying this algorithm to different predicates, makes possible a computation of the main rule-attribute, independently from the observations set. For instance, applying the algorithm with input predicate $p_1(x)$ gives the coverage of rule:

$r_1(x)$: if $p_1(x)$ then $q_1(x)$.

Now lets take a second rule:

$r_2(x)$: if $p_2(x)$ then $q_2(x)$.

Applying the algorithm with input predicate

$p_1(x)$ and $p_2(x)$ and not ($q_1(x)$ and $q_2(x)$)

yields the divergence between the two rules, that is an interval form representation of the set where both rules contradict.

Then, computing a rule's divergence with all the system's hidden rules, gives a universal set of counter-examples of that rule in interval form.

4.3 Measure of solution sets

On the list of intervals resulting from the interval-form algorithm, a measure can be define associating a real number to the solution set of the input predicate.

Using this measure the following rule attributes are computed:

a) The *utility* of a rule, that is the percentage of elements covered by the rule over the set of all elements.

b) The *invalidity* of a rule, that is the percentage of the rule's universal counter-examples over the set of all elements.

For example, the depth attribute in the physiotherapy environment has real values in the bounded interval [0,7], the frequency attribute has discrete values <0.75, 1, 3>.

For a rule like:

if depth(x) > 3.5 then frequency(x) = 0.75,

the interval-form algorithm will associate to the condition predicate the interval [3.5, 7] corresponding to the antecedent of the of the predicate. The measure of this interval is (7 - 3.5) while the measure of all the possible values is 7. For this rule, the utility will be

$(7 - 3.5) * 100 / 7 = 50\%$,

showing the rule has a good coverage and is quite useful, which doesn't mean of course it is valid.

On the other hand, computing with the interval-form algorithm the divergence of the rule with the system's hidden (valid) rules :

if depth(x)≤ 2.0 then frequency(x) = 3
if 2.0 < depth(x) < 4.5 then frequency(x) = 1
if depth(x) > 4.5 then frequency(x) = 0.75

yields the incompatibility interval list: ([3.5, 4.5], <0.75, 1, 3>). Measuring this interval list over the interval list of all possible values gives an invalidity measure of

$(4.5 - 3.5)*3*100/21 = 14.3\ \%$.

In other words, the rule is wrong 14.3 % of the time.

5. Embedding heuristic meta-knowledge in tools

With the interval-form algorithm and the different measures on the resulting solution-set the rule attributes lie on solid grounds. It is then possible to embed heuristic meta-knowledge in different tools, giving help to the learner in his rule induction activity, independently of his observation set.

In the physiotherapy environment, a set of about thirty meta-rules are associated with the prediction, validity, completeness and comparison tools. Each meta-rule uses the above rule attributes and others to capture properties of the student's knowledge base at any moment, and to build appropriate messages to the student.

Unlike the usual practice in Intelligent Tutoring Systems, the decision for advice display is up to the student. Each of the tool has a private menu in its window bar containing an option to ask for a methodological advice.

5.1 Heuristics for the validity tool

As a first example, let's look back at Figure-3 in section 2 presenting one use of the validity tool in the physiotherapy environment. This tool can list all the rules inconsistent with the selected rule Frequency-2 with the help the interval-form algorithm. Each rule having non-empty divergence with rule Frequency-2 is included in the list.

Using the suggestion facility, the student has triggered the following meta-rule:

```
(iff (and(=cex+ 0) (not (string=? chaîne)))
    (message
    "The rule is valid for all the cases but it
    is not consistent with other rules.
```

SUGGESTION: Keep the selected rule but test the validity of all inconsistent rules."))

This rule has fired because both of its conditions are met:

(= *cex*$^+$ *0)* means that the invalidity measure is 0 so there can exist no counter-example of the rule, even outside the small observation set the learner is currently using.

(not (string=? chaîne "")) means that the inconsistency list is not empty; there are rules in the same rule-group, inconsistent with the selected rule

5.2 Heuristics for the completeness tool

Lets look now at Figure-5 in section 2, showing the comparison Venn diagram between rule Duration-6 and rule Duration-5. The four percentages given on the lower part of the figure are computed with the measure defined on the interval form of each subset involved. From left to right:

- *rs-rc*: is the percentage of elements covered by rule-A but not rule-B;
- *inter*: is the percentage of elements covered by both rules;
- *rc-rs*: is the percentage of the cases covered by rule-B but not rule-A;
- *exter* = 100% - (rs-rc + inter + rc-rs) is the percentage of cases not covered by either rule.

Now in the case shown on Figure-5, we have rs-rc = 0 and rc-rs ≠ 0. The conditions of the following meta-rule are being met,

(iff (and (= rs-rc 0)(not(= rc-rs 0)))
(message
"Rule A is less general then Rule B. SUGGESTION: Verify the validity of this last rule. If it is, discard the less general rule A."))

When the student will ask for a suggestion, he will have the corresponding message displayed.

6. Conclusion: future developments.

The LOUTI design system components presented here can be labeled as a *computer-assisted induction system*. It is inspired by some aspects of the scientific discovery process [Holland et al. 1987; Langley et al. 1987]. This is why we have used terms like "observation set" or "prediction".
One way to extend the system is to enable it to tackle other forms of assertions used in scientific induction; for example, universally and existentially quantified relations between variables.
Another issue is generality. The rule component of the knowledge base and the associated domain are independent. For example, a similar environment has been experimented in an elementary school. With a slight adaptation to the interface, it has enabled pupils to write rule systems for translation from the Roman numeration system to the decimal system.
Another aspect of the generality problem is the rule predicates complexity. There is a trade-off between the user's freedom of expression and the degree of support by the system. We have decided to let the designer choose between two possibilities:

1- To limit the user to one "then" predicate of the form <attribute=value> and to an "if" predicate with a limited number of atomic clauses containing no functions. These limitations are needed for the interval-form algorithm to perform computations essential to many tools. Other limitations are also needed by interface considerations.

2- To have the student write complex predicates he chooses in the very general formal language used in LOUTI. In this option, some of the tools relying on the interval-form algorithm may not be available to the student.

Finally, beside its training applications, the system we have presented can be useful for expert-system development. The tools we have developed, and others we have in mind, are well adapted to view different aspects of a small set of rules, before they are embedded in a larger system. This is another extension of the present research we will explore in the coming months.

Acknowledgments

The authors are grateful for the support they has received from the other members of the GOIA and Micro-Intel research team. Anne Bergeron has shown the way with her previous prototyping of a knowledge-base for sets and relations processing. Serge Carrier and Annick Hernandez have programmed parts of the physiotherapy environment. Finally, Jacques Bordier and Charles Camirand made useful pedagogic comments for the system's improvement.

References

[Bergeron and Paquette 1989] A. Bergeron and G. Paquette, Discovery environments and Intelligent Learning Tools, in *Intelligent Tutoring Systems: At the Crossroad of Artificial Intelligence and Education*, Frasson C. and Gauthier G.(Eds), Ablex Publishing Co., 1990.

[Bergeron et al. 1988] A. Bergeron, L. Bouchard and R. Nadeau, A Multi-paradigm Development System For Exploratory Environments, *Proceedings of CSCSI 88*, Edmonton, 1988.

[Bordier and Paquette 1990] Jacques Bordier and Gilbert Paquette, Building Discovery Environments using Generic Software, paper accepted in *WCCE-90 Proceedings*, Elsevier, North Holland, 1990.

[Borning 1981] A. Borning, The programming language aspects of Thinglab, a constraint-oriented simulation laboratory, *ACM Transactions on Programming Languages & Systems 3*, pp. 353-387, 1981.

[Brown and Sleeman 1982] John S. Brown and Derek Sleeman, *Intelligent Tutoring Systems*, Academic Press, London, 1982.

[Clancey 1988] W.H Clancey and K Joerger, A Practical Authoring Shell for Apprenticeship Learning, *Proceedings of the ITS-88 conference*, Université de Montréal, Canada, June 1988.

[Ennals 1983] R. Ennals, *Beginning Micro-Prolog*, Harper and Row, New-York, 1983.

[Gaines 1988] B.R.Gaines, From teaching machines to knowledge-based systems: changing paradigm for CAL, *Proceedings of the international Conference on Computer Assisted Learning in Post-Secondary Education*, Calgary, Canada, 1988.

[Glaser et al. 1988] 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, Canada, 1988.

[Holland et al. 1987] J.H.Holland, K.J. Holyoak, R.E. Nisbett and P.R. Thagard. *Induction, Processes of Inference, Learning, and Discovery*. MIT Press, Cambridge, Mass., 1987.

[Langley et al. 1987] P. Langley, P., H.A.Simon, G.L.Bradshaw and J.M Zytkow, *Scientific Discovery, Computational Explorations of the Creative Processes*, MIT Press, Cambridge, Mass., 1987.

[Lawler 1987] R.W Lawler, Learning Environments: Now, Then and Someday, in Lawler, R. and Yazdani, M (Eds), *Artificial Intelligence and Education, vol. 1.* Ablex Publishing Co.,1987.

[Michalsky et al. 1983], R. S Michalsky et al. (Eds), *Machine learning I.* Tioga Pub. Co, 1983.

[Papert 1980] Seymour Papert, *Mindstorm: Children, Computers and Powerful Ideas.* Basic Books, 1980.

[Paquette 1988] Gilbert Paquette, Environnements d'apprentissage à base connaissances, Invited paper at the *International conference on Computers in schools. University of Bologna,* June 1988.

[Self 1990] John Self, Bypassing the Intractable Student Problem, in *Intelligent Tutoring Systems: At the Crossroad of Artificial Intelligence and Education*, Frasson C. and Gauthier G.(Eds), Ablex Publishing Co., 1990.

Figure Correctness in an Expert System for Teaching Geometry

Richard J. Allen
St. Olaf College Northfield, MN 55057 U.S.A.
Pierrick Nicolas
I.R.I.S.A. - I.N.S.A. Campus Universitaire de Beaulieu Rennes 35042 France
Laurent Trilling
I.M.A.G. - L.G.I. Grenoble Cedex 38041 France

Abstract

MENTONIEZH (the Breton word for *geometry*; a combination of MENT, *measurement*, and ONIEZH, *science of*) is an expert system designed to help secondary school students solve problems in geometry. The system is organized around four components, the first of which is figure construction by the student. The student uses the capabilities provided by the system to construct a geometric figure which conforms to a logical specification of the figure given by the teacher. The pedagogical objective of this component with its accompanying activities is to assure that the student has a solid understanding of the hypotheses involved in a problem before proceeding further with its solution.

Attempt is made to express criteria which assure that the construction of a figure conforms to a previously given logical specification of it. The approach taken is first to associate a logic formula to the constructed figure and another to the specification of the figure, and then to formulate the desired correctness as a relationship between these two formulas. Usually, a constructed figure must represent not just a particular case, but rather exhibit the generality intended by the given specification of it. Showing the figure is correct with respect to the specification seems simpler when a human does it than when one attempts to model this process. To obtain a satistfactory and decidable formulation of correctness, we use a method for constructing a well-founded extension of one fomula (the specification) with respect to the other (the figure). In this way, the required equivalence between a figure and its specification can be adequately expressed and verified.

1 Introduction

The system MENTONIEZH [Allen *et al.*, 1985; Nicolas, 1989] was conceived at I.R.I.S.A. as a help to secondary students in solving problems in geometry. It consists of four components:

- Figure Acquisition: the student constructs a figure which conforms to the teacher's specifications.
- Figure Appropriation: the student or the system itself can make graphical changes to a figure while preserving or not its logical properties. The student can discover interesting invariant as well as observe graphically the impact of suppressing a hypothesis.
- Property Exploration: the student expresses his opinion on possible interesting properties suggested by the system using theorems furnished by the teacher as aids.
- Proof Organization: aided by a list of relevant theorems provided by the teacher and stimulated by relationships and properties discovered during activities performed while using the three preceding components, the student constructs a proof which will be verified by the system.

This paper focuses exclusively on the first component of the system. Pedagogically, the figure acquisition component helps assure that the student has a solid understanding of the hypotheses of the problem. It does this using a graphics language which asks the student to "paraphrase" graphically the hypotheses of the problem which are normally presented to him in a natural language statement of the geometry problem under consideration. We believe that success at this activity indicates a reasonable understanding of what the problem's hypotheses are.

One question which arises here pertains to the criteria which allow verification that a student's graphical paraphrase (i.e., the class of figures characterized by his constructed figure) conforms to what the teacher has in mind (i.e., the specifications). Before addressing this question, we must first indicate the means given the student to construct a figure and the means given the teacher for specifying the intended class of figure. For constructing geometric objects, our interface provides a language with simple drafting capabilities (e.g., objects constructed using pencil, ruler, compass, protractor). Another language, a formal language similar to those utilzed in classroom geometry textbooks, is provided for specifying objects and logical relationships among them. Synopses of these two languages are presented in section 2 and 3, respectively.

Finally, a third language is defined for expressing precisely the correctness of a constructed figure with respect to a given specification. This language which is summarized in section 4 is a logic-based language and is used to construct a formula F (for Figure) to describe the student's figure and to construct another formula S (for Specification) to represent the teacher's specification. A set of axioms, called a TIG (Théorie Instrumentale de la Géométrie), is associated with this language and encapsulates the basic geometric knowledge that the teacher assumes students to have.

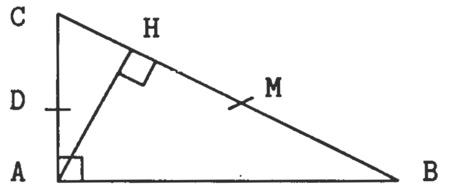
Section 5 contains the criteria that the desired correctness must satisfy. On the one hand, the teacher wants the student to construct the most general type of figure satisfying the specifications. This means the student's figure satisfies all logical properties given in the specifications but not additional logical properties not found in the speccification. For example, if the specifications indicates just two lines intersecting, one doesn't want the student to draw them perpendicular to one another; one really wants a certain kind of equivalence between figure and specifications. On the other hand, one cannot exclude *a priori* the student from creating in his figure more geometric objects (supplementary ones) that are given explicitly in the specification. The situation is further complicated by the need to remain, for reasons of decidability, in a finite universe of objects. In section 6, we show satisfaction of the criteria by creating an extension of the specification formula as a function of the construction.

In this paper, our focus is not on automatic theorem proving [Gelernter *et al.*, 1963; Coelho and Pereira, 1979] nor on aiding problem solving in geometry [Anderson *et al.*, 1985; Chouraqui and Inghilterra, 1987; Py, 1990]. Rather, we are trying to make more precise what a teacher means when he says a constructed figure is a correct paraphrase of what a problem statements intends.

2 SCL (Student Construction Language)

This language provides the interface with which a student (1) indicates on a menu what object (point, line, ray, segment, circle) he wants to draw on the graphic tablet; (2) draws the object; and (3) expresses the logical properties (name, belonging to another object, parallel to, perpendicular to, ...) of the drawn object with respect to already-drawn objects. Details concerning both the functionning of the system and its logical capabilities can be found in [Nicolas, 1989]. What is important to point out here is that each operation must be realizable using drafting table instruments (ruler with translation, square, compass). Consequently, the student is limited in his constructions. For example, he cannot construct a line passing through three points although he could create a line and afterwards three points belonging to it.

As an example, consider the statement: Let ABC be a right triangle at A. Let H determine the height coming from A and let M and D be the midpoints, respectively, of the sides BC and AB. [Students are usually asked to show there is a circle passing through A, H, M and D given the preceding statements as hypotheses.]



A possible construction by the student might be encoded using the following sequence of steps:

OBJECTS and PROPERTIES
point name A
point name B
segment with endpoints A and B
segment with endpoints A and C with support perpendicular to support of preceding segment
segment with endpoints B and C
line passing through A and perpendicular to support of preceding segment
point name H belonging to line (BC) and to preceding line
point name M belonging to segment [BC] and at distance from B equal to one-half |BC|
point name D and belonging to [AB] at distance from A equal to one-half |AB|

3 CDL (Classroom Description Language)

This language is similar in syntax to those used in textbooks [Deledicq *et al.*, 1983] to decribe geometric properties and is to be used by the teacher to specify the hypotheses of the problem. Basic elements are given below.

TERMS:
x,y,... (lower cases): identifiers designating unnamed objects
A,B,... (upper cases): identifiers designating named objects
(pq): line passing through points p and q
|pq|: distance between points p and q
2d, 1\2d: twice, one-half the distance d
[pq): ray with origin p passing through q
[pq]: segment with endpoints p and q
centre(c): center of circle c
rayon(c): radius of circle c

PREDICATES:
point(p): p is a point
droite(l): l is a line
demi-droite(h): h is a ray
segment(s): s is a segment
cercle(c): c is a circle
distance(d): d is a distance
x = y: x and y are the same objects
x ∈ y: point x belongs to y

l1 || l2: lines l1 and l2 are parallel
l1 ⊥ l2: lines l1 and l2 are perpendicular
memesens(h,h'): rays h and h' have the same direction
invsens(h,h'): rays h and h' have opposite directions
d1 < d2: distance d1 is less than distance d2
d1 > d2: distance d1 is greater than distance d2

Example: The teacher's specification for the figure to be constructed in the preceding example could be given in CDL as a conjunction of the following atomic formulas:

$$[AB] \perp [AC],\ (AH) \perp [BC],\ H \in (BC),\ M \in [BC],$$

$$|BM| = |MC|,\ D \in [AB],\ |AD| = |DB|$$

4 LDL (logical Description Language)

This language provides a commom medium in which both the figure and the specification can be represented. It contains six type predicates and ten property predicates which are described below.

TYPES:
point(p): satisfied if p is a point
droite(l): satisfied if l is a line
demi-droite(h,o,l): satisfied if h is a ray with origin o and support l
segment(s,p1,p2,l): satisfied if s is a segment with endpoints p1 and p2 and with support l
cercle(c,o,r): satisfied if c is a circle with center o and with radius r
distance(d): satisfied if d is a distance

PROPERTIES:
appdr(p,l): satisfied if the point p belongs to the line l
appdd(p,h): satisfied if the point p belongs to the ray h
appseg(p,s): satisfied if the point p belongs to the segment s
appcc(p,c): satisfied if the point p belongs to the circle c
memesens(h1,h2): satisfied if the rays h1 and h2 have the same direction
invsens(h1,h2): satisfied if the rays h1 and h2 have opposite directions
par(l1,l2): satisfied if the lines l1 and l2 are parallel
perp(l1,l2): satisfied if the lines l1 and l2 are perpendicular
demidist(d1,d2): satisfied if the distance d1 is equal to one-half distance d2
infdist(d1,d2): satisfied if the distance d1 is less than the distance d2

The figure construction previously described using SCL could be associated with the LDL formula F:
point(A) point(B) point(C) point(H) point(M) point(D)
droite(l1) droite(l2) droite(l3) droite(l4)
distance(d1,B,M) distance(d2,B,C) distance(d3,A,D) distance(d4,A,B)
segment(s1,A,B,l1) segment(s2,A,C,l2)
segment(s3,B,C,l3)
appdr(H,l3) appdr(H,l4) appdr(A,l4)
appseg(M,s3) appseg(D,s1)
perp(l1,l2) perp(l3,l4)
demidist(d1,d2) demidist(d3,d4)

The specification previously described in CDL could also be associated with the LDL formula S:
point(A) point(B) point(C) point(H) point(M) point(D)
droite(l) droite(l') droite(l") droite(l"')
distance(d,B,M) distance(d,M,C) distance(d',A,D) distance(d",D,B)
segment(s,A,B,l) segment(s',A,C,l') segment(s",B,C,l")
appdr(H,l") appdr(H,l"') appdr(A,l"')
appseg(M,s") appseg(D,s)
perp(l,l') perp(l",l"')

F is obtained by associating with each object as it is being constructed an object predicate and one or more property predicates. In a similar way, S is constructed by associating with each object from the CDL notation a type predicate and property predicates which correspond to the explicitly expressed logical relations.

5 Figure Correctness Criteria with respect to a Specification

There are four such criteria. Each criterion has associated with it a key question (and in one case, two questions), the answer to which allows one to determine whether or not the criterion is satisfied.

(1) Graphical Correctness: Does the drawn figure contain all the geometric objects intended by the teacher in his specification?

Our resolution to this question is simple: the figure must contain all objects on which the specification operates. However, one must be cautious as to what is implied in the construction of composed objects. In the preceding example, the specification in CDL contains both [BC] and (BC). Geometrically speaking (BC) is a component of [BC]. In LDL, the use of the predicate segment(s",B,C,l") expresses explicitly that the line l" associated with (BC) belongs to the term representing [BC]. Consequently, if the student first constructs the segment [BC], (which is translated by the predicate segment(s3,B,C,l3) in F), then he will not have to construct the line (BC). On the other hand, the reverse is not true. Even if in a specification one finds both [BC] and [BC), the construction of [BC] beforehand does not assure that of [BC) since a ray is not a parameter of segment(s,p1,p2,l).

(2) Consistency of figure F and of specification S: Are F and S coherent?

One must be sure that there are not terms in S that could imply contradictory information. For example, if Euclid's axiom is allowed, one must exclude a specification for which the corresponding LDL formula S implies that two distinct lines which intersect in a point are parallel. One must also be sure that the LDL formula F corresponding to the construction is not contridactory. Of course, regardless of how accurate the graphics are, it is always possible for the student to draw, incorrectly, two lines parallel to a third line and passing through a common point.

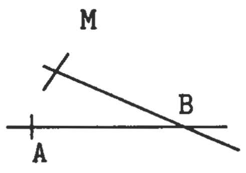
(3) Constructibility: Does at least one figure exist that can be constructed and that is correct with respect to the specification?

There are specifications for which there are no figures constructible with ruler and compass. For example, the trisector of a 60° angle is defined by the specification:
O = centre(C), M1 ∈ C, M2 ∈ C, M3 ∈ C, M4 ∈ C
|M1M4| = rayon(C),
|M1M2| = |M2M3| = |M3M4|
yet is not constructible with ruler and compass.

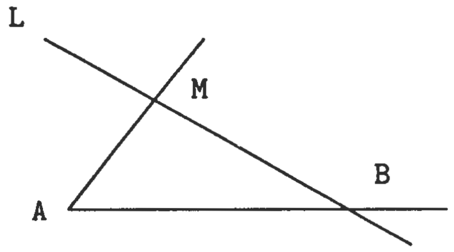
(4) Correctness of F with respect to S: (i) Does F satisfy the conditions specified in S? (ii) Is F not particular?

This criterion is complicated by the fact that it is unreasonable to prohibit a student from constructing more objects than are used in S. For example, consider the specification S and two constructions F and F', all three encoded in LDL:

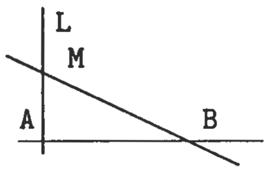
S ≡ A ∈ l1, B ∈ l1, M ∈ l2, B ∈ l2, M ≠ A



F ≡ A ∈ l1', B ∈ l1', M ∈ l2', B ∈ l2', A ∈ L, M ∈ L, M ≠ A



F' ≡ A ∈ l1', B ∈ l1', M ∈ l2', B ∈ l2', A ∈ L, M ∈ L, M ≠ A, L ⊥ l1'



Note that F satisfies both 4(i) and 4(ii), whereas F' satisfies only 4(i) and not 4(ii). However, both F and F' contain more objects than are referred to in S.

6 Logical Expression of the Correctness Criteria

In this section issues raised by each question related to the correctness criteria are elucidated. The order of treatment of the questions is different than in section 5, beginning with the consistency criterion.

Consistency. Here we need to introduce a set of axioms (TIG) representing the student's basic knowledge in geometry. Then consistency of F and of S is expressed by two conditions:

(I) TIG,S $\nvdash \square$
(II) TIG,F $\nvdash \square$

where □ represents the empty clause.

In order for (II) not to cause rejection of F as containing contradictory information when, in fact, it does not, we must choose TIG that it is contained in the actual theory of geometry being used in classrooms. [It would be easy just to verify individually that every axiom of TIG is a correct formula of geometry.]

In order for conditions (I) and (II) to be effectively verified, TIG must be decidable. And, in order for TIG to be modified and calibrated to the level of the student, restriction that bear upon it must remain simple. We have chosen the restriction of Bernays-Schonfinkel [Ackerman, 1954], which provides for universally quantified formulas not containing function terms (the Herbrand universe remains finite). For example, an axiom such as:

$$\forall\, x, P(x) \Rightarrow (\exists\, y, Q(y))$$

is not allowed (skolemization would introduce function terms).

Constructibility. To show that there exists (or not) a figure constructible with ruler and compass satisfying a given specification is a difficult problem [Carrega, 1989], but a decidable one [Lebesgue, 1989]. However, in our case, rather than employ complicated means, it seems preferable in our teaching context to ask the teacher who has drawn up the specification to put himself in a student's place and to provide at least one correct construction. However, note that even if SCL can be used to produce a drawing, thereby almost certainly guaranteeing a construction by ruler and compass, this does not constitute a formal proof that a figure can be constructed by ruler and compass.

Figure Correctness with respect to the Specification and Graphical Correctness. Let S(a,a',...) denote the Formula S where a, a',... are the unnamed objects of S. Let OF be the set of objects of F. Criterion 4(i) is expressed by:

TIG, F ⊢ ∃ a,a',... (S(a,a',...), a ∈ OF, a'∈ OF,...)

Note that this formulation takes into account a part of the graphical correctness criterion (1). In fact, the formulation requires that all the objects present in S also be found in F and, therefore, have been constructed by the student.

Observe that the unnamed objects of S must associated with distinct objects in OF – ONS, where ONS is the set of named objects of S. For example, consider a specification for a triangle ABC with heights h1, h2 and h3 emanating respectively from A, B and C, and with the points of intersection of h1 and h2 and of h2 and h3. It is clear that a correct construction must contain the creation of these two intersection points (even if geometrically they are the same). Conditions such as these can be verified easily by letting the predicate Tousdifférent(a,a',...) model the fact that a, a',... are all syntactically different. Then correction criteria 4(i) and (1) become

(III) TIG, F $\vdash \exists$ a,a',... (S(a,a',...), a $\in$ OF − ONS, a'$\in$ OF − ONS,... Tousdifférent(a,a',...))

Particularity. Establishing a criterion associated with this question is complicated by two requirements. On the one hand, one cannot forbid the student from creating more objects than are given. On the other, the axioms of TIG are in a form such that the universe of objects is fixed (being the set of OF).

Consider μ(S) where μ is a substitution {a=..,a'=..,...} such that (III) is satisfied. We seek an etension S^* of μ(S) with respect to F by adding geometrically reasonable properties to μ(S) for objects belonging to OF − OS. S_0, S^* and F operate on the same vocabulary of objects. Intuitively, adding a geometrically reasonable new object to a formula comes down to adding to the formula those minimal properties corresponding to a geometric-instrument construction of the object in relation to objects present in the formula. Then correction criteria 4(ii) becomes

(IV) TIG, S^* $\vdash$ F if there exists an extension S^* of μ(S) with respect to F.

Example: Recall the specification S and the two figures F and F' presented in the example found at the end of Section 5. Let μ = {l1=l1', l2=l2'} and define $S^* \equiv$ μ(S), droite(L), A $\in$ L, M $\in$ L.
S^* and F satisfy (IV) yet S^* and F' do not satisfy (IV). Moreover, μ(S) admits two other simple extensions S^{*}' and S^{*}" with respect to F' and neither of these satisfies (IV):

$$S^{*}\text{'} \equiv \mu(S), \text{droite}(L), A \in L, L \perp l1'$$
$$S^{*}\text{"} \equiv \mu(S), \text{droite}(L), M \in L, L \perp l1'$$

The construction of an extension S^* is defined using so called "extension axioms" of the type: $\forall$ y1,y2,..., CE(y1,y2,...) $\Rightarrow$ ($\exists$ x, Propmin(x,y1,y2,...)). For each type of object x, one can introduce such axioms to model the unique construction of x using the construction instruments of geometry and constructing from the objects y1,y2,... . For example, associated with the construction of a line, one has the extension axiom:

(*) $\forall$ p1,p2 p1$\neq$p2 $\Rightarrow$ ($\exists$ l, droite(l), p1$\in$l, p2$\in$l)

The extension S^* is then constructed using the sequences ES_i and OS_i such that:
$ES_0 \equiv \mu(S)$
$ES_{i+1} \equiv ES_i$, Propmin$_i$
$ES_n \equiv S^*$
OS_0 = OS
$OS_{i+1} = OS_i \cup ob_i$
OS_n = OF
where Propmin$_i$ and ob_i are determined in the following two ways:

(i) if there is $o_i \in$ OF − OS_i and if there is an extension axiom such that (a) and (b) are satisfied:
(a) ES_i, R(o_i) $\Rightarrow$ ($\exists$y1,y2,..., Propmin(o_i,y1,y2,...)) where R(o_i) is the conjunction of all litterals of F where o_i appears alone or with some object belonging to OS_i.
(b) $ES_i \Rightarrow \mu_i$(CE(y1,y2,...)) where μ_i = {y1=.., y2=.., ...} satisfies (a) then Propmin$_i$ = μ_i(Propmin(o_i, y1, y2,...)) and ob_i = {o_i}.
Example: Let us return to the last example and extension axiom * above. Then, using *, we have:
o_0 = L
$ES_0 \equiv \mu(S)$
R(o_0) $\equiv$ droite(L),A $\in$ L, M $\in$ L
μ_0 = { p1 = A, p2 = M }
CE(p1,p2) $\equiv$ p1 $\neq$ p2,
μ_0(p1 $\neq$ p2) $\equiv$ A $\neq$ M
Propmin(L,p1,p2) $\equiv$ droite(L), p1 $\in$ L, p2 $\in$ L.
Propmin$_0$ $\equiv$ μ_0(Propmin(L,p1,p2)) $\equiv$ droite(L) A $\in$ L, M $\in$ L.
Since OS_1 = OF, we obtain

$$S^* \equiv \mu(S), \text{droite}(L), A \in L, M \in L.$$

The extension S^{*}' and S^{*}" with regard to F' are obtained by considering an extension axiom signifying that there exists a unique line passing through a point and perpendicular to a line (no matter what the point and the line may be).

(ii) For no element of OS_i does there exist an extension axiom such that (a) is satisfied. This signifies that no object of OF − OS_i is uniquely constructed from the objects of OS_i. One can then add all the properties of these objects to ES_i; so let

Propmin(o_i $\equiv$ conjunction of all predicates of F depending on objects belonging to OF − OS_i
OS_{i+1} = OF.

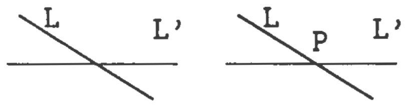
Note that (1) can be satisfied without (2) being satisfied. In this case, one cannot construct an extension and F is not acceptable. For example, let

S $\equiv$ droite(L), droite(L')
F $\equiv$ droite(L), droite(L'), P $\in$ L, P $\in$ L'

with extension axiom

$\forall$ l, l', $\neg$(l||l') $\Rightarrow$ ($\exists$p,p', p$\in$l, p'$\in$l').

Since $\neg$(L||L') cannot be deduced from S, the construction F is not acceptable.



7 Conclusion

We have identified four conditions for expressing the criteria of figure correctness with respect to a specification. It is natural to look at the characteristics of the set of axioms of TIG in the context of the student knowledge TIG represents and from the point of view of modifications the teacher might wish to make. In addition, we need to look at the problems involved in implementing the verification of these four conditions.

The first question is how powerful the set of axioms TIG should be. It seems desirable to be able to detect and point out reconstruction of clearly useless objects and redundant definitions. For example, if a student has constructed a segment [AB] and he then constructs the line (AB), it is desirable to indicate to him that the second construction is unnecessary. In addition, if the teacher has named M (resp. M') as the intersection point of h1 and h2 (resp. h2 and h3), then it is undesiderable to allow TIG to permit a construction where the student indicates M (resp. M') is the intersection point of h2 and h3 (resp. h1 and h2).

We have manually identified about forty axioms as a base, called TIG_0, from which we hope a useful, robust TIG can evolve. Our method has consisted in itemizing all the apparently elementary cases in which the truth of a predicate can be implied. We have an approach *à la Horn*; however, some clauses contain negations. With TIG_0 it is not possible to cause the deduction M = M' in the preceding example. However, the problem of calibrating TIG remains very difficult especially since we lack of a complete theory of geometry encoded in first order logic terms with predicates similar to those in LDL. Our situation is different from that in [Hilbert, 1971] and in [Lelong-Ferrand, 1985]. There is an axiomatic formulation in [Tarski, 1959], but it is not very helpful since it uses only three predicates. However, in-depth study of this axiomatization seems indispensable if one wants to construct a complete TIG.

A second problem concerns the adoption, or not, of the unique-name assumption for S as well as for F. For example, if $S \equiv A \in L, B \in L, A \in L', B \in L'$
and if the axiom of the uniqueness of a line passing through two distinct point is part of TIG, then either $A \neq B$ and $L = L'$, or $A = B$. Is S acceptable under such conditions?

What makes us lean toward adoption of the unique-name assumption is the desire to respect graphical correctness. If the student must construct all objects in S and if one wants to recover all constructions made, then the objects of S must be different. If view of this, (I), (II), (III) and (IV) must all be understood with this assumption in force. Consequently, the preceding specification would be rejected.

The implementation already completed [Nicolas, 1989] contains translations SCL to LDL and CDL to LDL and verification of (III), the latter of which is achieved by a saturation method which has succeeded a prohibitively high costs (on the order of 20 minutes on a SUN3 for the example given in Section 2). We are currently investigating how to represent TIG_0 in a logic program with reasonable simplifications so that one can obtain acceptable implementation running costs.

The extension used in (IV) completes μ(S) with objects from F. Making use of it calls into play two levels of nondeterminism: both μ and each ES_i can be determined in several ways. The first is not too costly, especially if the teacher names most of the objects he uses. The second poses more of a problem because failure to satisfy (IV) can result from the non-existence of the extension, which necessitates trying all possible extensions. However, a promising new way to detect these failures resides with constructing the ES_0, ..., ES_n by following the order in which the student constructs his objects.

It is important to underscore how the finite universe of objects is constructed here. On the one hand, it is done through translation of SCL to LDL and CDL to LDL. The choices of objects made (e.g., creation of the support (AB) if the segment [AB] is created but the distance |AB| is not) clearly have implications for the deductions possible and can be reevaluated after some experimentation. On the other hand, the set of extension axioms is crucial for obtaining a good modelling of what happens in geometry.

References

[Ackerman, 1954] W. Ackerman. *Solvable cases of the Decision Problem.* North Holland, Amsterdam, 1954.

[Allen *et al.*, 1985] R. Allen, P. Nicolas, and L. Trilling. Learning Geometry with the Assistance of Logic Programming. In *Supporting Papers of the ICMI Symposium on "The Influence of Computers on Mathematics and its Teaching"*, pages 131–137, Strasbourg, 25–30 March 1985.

[Anderson *et al.*, 1985] J.R. Anderson, C.F. Boyle, and G. Yost. The Geometry tutor. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 1–7, Los Angeles, California, August 1985.

[Carrega, 1989] J.C. Carrega. *Théorie des corps, la règle et le compas, réédit.* Herman, Paris, 1989.

[Chouraqui and Inghilterra, 1987] E.Chouraqui and C. Inghilterra. Apport de la méthodologie fondée sur les objets pour la conception d'un système d'E.I.A.O. de la géométrie. In *Proceedings of COGNITIVA87*, pages 39–44, Paris, May 1987.

[Coelho and Pereira, 1979] H. Coelho and L.M. Pereira. A PROLOG Geometry Prover. Technical Report 525, Laboratorio Nacional de Engenharia Civil, Lisbon, 1979.

[Deledicq *et al.*, 1983] A. Deledicq, C. Lassave, C. and D. Missenard. *Mathématiques quatrième.* Cedic-Nathan, Paris, 1983.

[Gelernter *et al.*, 1963] H. Gelernter, J.R. Hansen and D.W. Loveland. Empirical Exploration of the Geometry Theorem Proving Machine. *Computers and Thought*, E. Feigenbaum and J. Feldman, McGraw-Hill, New York, 1963.

[Hilbert, 1971] D. Hilbert. *Les Fondements de la Géométrie, édition critique préparée par R.Rossier.* Dunod, Paris,1971.

[Lebesgue, 1989] H. Lebesgue. *Leçons sur les constructions géométriques, réédit.* Gauthier-Villard, Paris, 1989.

[Lelong-Ferrand, 1985] J. Lelong-Ferrand. *Les Fondements de la Géométrie, réédit.*. Presses Universitaires de France, Paris,1985.

[Nicolas, 1989] P. Nicolas. Construction et Vérification de figures géométriques dans le système MENTONIEZH. Thèse de l'Université de Rennes1, 1989.

[Py, 1990] D. Py. Reconnaissance de plan pour l'aide à la démonstration dans un tuteur intelligent de la géométrie. Thèse de l'Université de Rennes1, 1990.

[Tarski, 1959] A. Tarski. *What is Elementary Geometry?* In *The Axiomatic Method, with Special Reference to Geometry and Physics*, L. Henkin, P. Suppes, and A. Tarski eds, North Holland, Amsterdam, 1959.

Improving the Quality of Case Memory Using Genetic Techniques

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

Abstract

Human problem solving uses past experiences to solve current problems. When faced with a problem, one often locates experiences (cases) in memory that are similar and adapts them to meet the current situation. The reasoning process that combines these tasks for problem solving is called Case-Based Reasoning (CBR). We also need to forget some experiences. Therefore, weak, invalid, and harmful cases must be removed from the case library, while strong ones remain. We have extended genetic algorithm techniques to apply to CBR. Using modified genetic techniques, we propose a method of learning cases in a noisy environment and controlling the integrity of case memory, and a method of generating novel cases that does not rely on the failures of other cases for the construction of a new case. We have tested our approach in the domain of a 4x4 checker game. Favorable empirical results, as predicted by genetic theory, have been achieved.

1 Introduction

To learn, a Case-Based Reasoning System (CBRS) [Kolodner, 1987; Kolodner, 1988] must be able to acquire and incorporate new and modified cases into its existing case library. New cases are typically generated by three methods: hand-coded cases from experts, cases adapted from existing ones by the CBRS, or newly developed cases produced by the CBRS that reflect the current environment better than any existing case in the library. One might believe that experts are best at generating cases representing solutions of a problem space. However, as [Bradtke, 1988] points out: 'the effectiveness of a case-base is the number of unique problem states underlying the case-base encoding', and humans tend to cluster cases about points in a problem space rather than providing unique points. Initial cases developed by experts - as opposed to cases that constitute an expert's solution to a specific problem posed by the CBRS - are more redundant than might be expected. CBR can also lead to redundancy, when modifications to a case and additions to the case library are very similar to existing cases.

The positioning of a case is not an easy task. For each newly generated case, the CBRS must consider whether it or a similar case already exists in memory, and thus should not be added at all, or, if it is added, how its addition will affect the existing cases in memory. For instance, the introduction of a new case may make some cases unretrievable or create opposing solutions to a problem.

There is another factor that can affect the relationships between the cases in memory. In a dynamic environment, changing conditions may render past solutions inapplicable to similar problems in the present. Memory, in effect, is growing old.

A CBRS must continually look for and remove redundant and possibly harmful cases, and must be able to cope with the three types of environmental changes: an anomaly (freak occurrence), an unforeseen but lasting change in the environment, and a probabilistic failure (predictable). Figure 1.1 helps to illustrate the different types of cases that can arise in memory due to environmental changes. Unique cases and loosely similar ones are desirable because they provide solutions to different problems in the problem space. Redundant and harmful cases are undesirable and need to be removed from memory because they do not lead to new solutions and may provide incorrect ones.

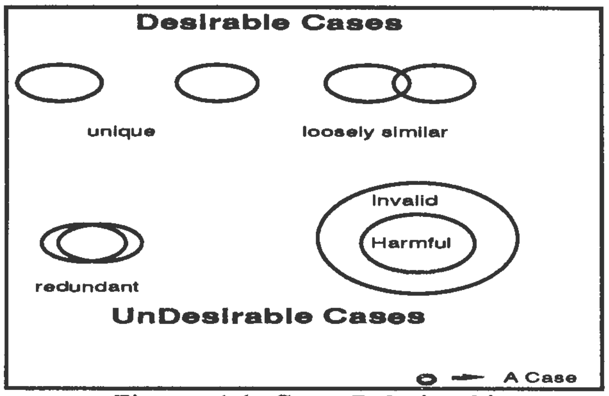


Figure 1.1 Case Relationships

Novelty, although not often part of a CBRS, can play an important part in the learning process. When one is solving problems using cases of past solutions, the solutions tend to become repetitive - stuck in a rut. Even the strongest of adaptive methods can make only similar adaptations of a case, time and time again, using the traditional methods of adaptation and analogy. What is more desirable is a method of altering a case in a novel way so as to, more often than not, produce a case that is not one of the normal adaptations [Sycara, 1988] of the case. We are looking for novel cases, loosely based on the original, but not an adaptation that is consistently performed using other traditional methods. If the problem of removing poor cases were solved, the system would be able to garbage collect them, removing unacceptable novel cases that were created and added to the case library. Good novel cases, however, will remain in the case library and improve its overall quality.

To address these problems, i.e. the production of radically novel cases and the detection and removal of redundant and invalid ones, we propose the use of a genetic algorithm [Goldberg, 1989], specifically, the application of a *Conservative Reproduction Algorithm for Cases* (CRAC), to handle the entropic graying of case memory. Genetic algorithms (GAs) simulate mechanisms of biological evolution, and can benefit case-based approaches in a number of ways through the application of their genetic operators.

In the following sections, we describe a combined genetic and case-based learning approach. Section 2 is dedicated to explaining how *a case is analogous to a chromosome string,* allowing us to combine the genetic and case-based techniques. Section 3 shows how the genetic concepts of reproduction and credit apportionment can be achieved with cases, giving rise to the CRAC, a method of generating a new case library, removing bad cases, and keeping good ones. Section 3 also describes the other genetic operators of mutation, crossover, division, and connection for use with cases. Section 4, by way of an example, describes how the genetic and case-based techniques are molded together to form our learning approach. Section 5 describes our results, and section 6 concludes with a brief summary.

2 Cases as Chromosome Strings

The notion of using GAs with cases differs from the traditional use of GAs in several respects. GAs typically encode the data as fixed length strings using the symbols 0 and 1 as the language of the alleles. This makes for ease of use in applying genetic operators; however, the expressive power is not enough for symbolic problem solving used by many expert systems today. It is a difficult task to encode values and all other parameters that contribute to a problem's description into a fixed length string of zeros and ones. A language of two symbols can only encode two possibilities for any one allele and is inadequate. Case structures, for example frames, use a rich symbol notation; thus, frames can represent a considerably larger number of values than that of a single allele. We increase the descriptive power of cases by increasing the alphabet, at the cost of increasing complexity.

Another important distinction is that cases do not necessarily have to have a fixed number of frames [Deugo, 1989a]. For example, a checker strategy could be considered as a case where each frame provides the next game move of the strategy. As a strategy grows and shrinks, so too does the number of frames in the case. Strings have a fixed size, so expansion or contraction of the number of alleles is impossible. Having dynamic cases expands the number of representations the case can form; thus, increasing the potential use of GAs. Often a problem produces more and more recognizable symptoms as it is better understood. Having dynamic cases provides a means of easily adding information to them.

To use a GA with cases, case frames require a strength field. Figure 2.1 outlines a general case frame structure to support a GA. Making the strength of a case a function of the strength of each frame, the addition of new frames causes no problem for the concept of strength.

The final distinction between a GA for cases rather than strings, is that a case is symbolic while a string is subsymbolic. It is easy to perform mutations and crossover on strings with 0 and 1, but the language is restrictive. As will be described later, genetic operators such as crossover and mutation can also be performed on cases. The process of using GAs with symbolic rules has been suggested [Bickel, 1987; Antonisse, 1987] and performed using bit strings to represent symbolic rules [Holland, 1986], but never have GAs been used with cases. The power of GAs makes them very attractive when applied to cases.

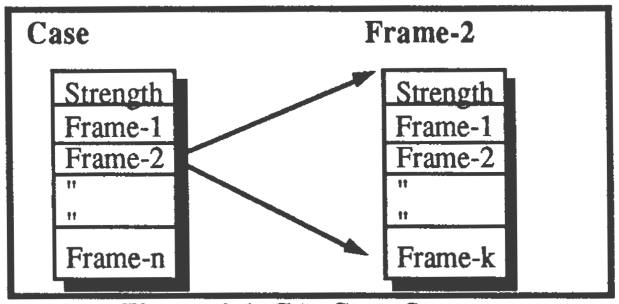


Figure 2.1 GA Case Structure

3 Case Library Reproduction

The solution to the main goal of maintaining a minimal set of strong cases in a fixed size case library is achieved with the *Conservative Reproduction Algorithm for Cases,* shown in Figure 3.1.

The traditional genetic reproduction operator selects a new population of chromosomes from the strongest chromosomes of the previous population. The CRAC reproduces and ensures that the next generation of the case library contains a nonredundant majority of strong cases from the past generation, but still allows for the introduction of new cases that have yet to be proven or disproven.

The CRAC contains three parts: case strength ordering, case selection, and new case generation. Memory is always a fixed resource; therefore, as input to the algorithm an upper bound on the number of cases in the library is provided. This is not an absolute upper bound because a percentage of that number in new cases may be added to the library, but for now it is reasonable to think

```
Input:  Case-Library, Minimum-Acceptable-Strength,
                Maximum-Number-Of-Cases, New-Case-Percentage
Output  New-Case-Library
Begin
        number-of-cases := 0
        New-Case-Library := Empty-Case-Library
        { Sort the cases into descending order based on their strength}
        ordered-case-library := Order-Case-Descending-In-Strength(Case-Library)
        { Select the maximum number of cases allowable for the next generation whose
                strength is above an acceptable value }
        current-case := Next( ordered-case-library)
        While ((current-case.Strength >= Minimum-Acceptable-Strength) AND
                (number-of-cases <= Maximum-Number-Of-Cases)) Do
                Add( current-case, New-Case-Library)
                current-case := Next( ordered-case-library)
                number-of-cases := number-of-cases + 1
        EndWhile
        { Normal CBR adaptation methods could be performed here if so desired}
        {Apply other genetic operators to probabilistically selected cases,
                adding a given percentage of new cases into the case library - see the next
                section for the details.  }
        For (size(New-Case-Library) * New-Case-Percentage) Times Do
                new-case := Probabilistically-Select-Case(ordered-case-library)
                new-case := Apply-Other-Genetic-Operators-To( new-case)
                Add( new-case, New-Case-Library)
        EndFor
End
```

Figure 3.1 Conservative Reproduction Algorithm For Cases

of it as an upper bound. Another parameter provided to the algorithm is a minimum acceptable strength threshold value. This value determines the minimum strength of a case that is permissible in the new case library. Our conservative approach assures that from one generation to another, a constant number of strong cases are always reproduced, yet it avoids duplication and guarantees the consistency and integrity of the library between generations. It does not, however, select the same cases from one generation to another. It is the performance of a case that indicates its acceptability for reproduction into the next generation. From an old case library a new one is produced with the weak cases weeded out, a small portion of new cases introduced, and a solid base of strong cases remaining.

3.1 Case Operators

The reproduction operator is only the first piece of a genetic algorithm; after selecting strong cases for the new library generation, other genetic operators are performed on selected strong cases providing a set of possibly novel cases for the next generation of the case library.

We have used four other case operators in addition to the CRAC: mutation, crossover, division, and connection. Mutation of a case is the process of changing the contents of one or more of its frames (slots) from their current values to another legal value. Crossover of two cases is the process of merging a collection of the first half of a case's frames with that of the second half of another case's frames. Division of a case is the process of splitting a case at a frame producing two smaller cases. Connection of cases is the process of joining two smaller cases producing a larger single case. Used in coordination with the division operator, the pair have the ability to produce strong new cases built from smaller subcases. Operators may be forced to make certain assumptions about the domain. For example, the Connection operator must be able to know that it is allowable to connect two pieces at a given point; however this complication increases the overall power of the operator and is worth the added trouble.

4 Our Approach

In this section, by way of an example, the complete genetic process for learning strong new cases is molded together. The process has two major steps: case reinforcement, and the application of the CRAC. Case reinforcement - the process of increasing or decreasing a case's strength - is performed by the modified Bucket Brigade Algorithm [Goldberg, 1989]. The algorithm uses the success or failure of the application of a case in its judgment to reward or penalize a case's strength. After a period of case reinforcement, the CRAC, using the genetic operators of mutation, crossover, division, and connection, reproduces the strong cases into the next generation of the case library and removes the weak,

invalid, and harmful cases. The learning process is an infinite cycle of using the cases, reproducing the best ones, and adding new cases formed from genetic adaptations of existing strong cases. The end result is a library of strong, non-redundant cases.
The example domain we will use is that of checkers. The initial goal is to produce a system that plays a game of checkers with a human and improves over time. The only piece of domain information the system begins with is what a legal move in a checker game is. Cases are used to represent checker move strategies; the CRAC, genetic operators, and the modified Bucket Brigade Algorithm are used to introduce and control the knowledge in the case library.

The goal is to show that using only a very small amount of domain knowledge, the system can generate and keep strong new cases for application in a checker game. As a measure of this, the average strength of a case should increase as the number of generations of the case library increases [Deugo, 1989b].

4.1 Structures

The basic and only structure is a case; a case defining a checker strategy. Reviewing the case structure described in section 2, a case has a collection of frames and a strength value; and a frame has a strength value and a data item. A checker case frame has a strength value and a checker piece's move as its data item. A move consists of a board layout and a legal move for a player. A checker case consists of a strength value and a set of checker frames - each frame a checker move - alternating between between legal moves of the white and black players in the game of checkers. A checker case - known as a strategy - is just a sequence of moves experienced during the progress of a checker game.

An important feature of a case is that its size is dynamic. A case strategy does not represent the moves taken in a full game of checkers, but rather that of a partial sequence of moves of different lengths. A good case can then be defined as a sequence of moves that leads the system player, black, to a better position.

New cases are formed in two manners: by the application of genetic operators, or by system creation. It has been described how genetic operators form new cases. Strong existing cases are taken and altered by the four genetic operators: mutation, crossover, division, and connection. However, before these operations can be performed, cases have to exist. This is where the domain theory, i.e. the legal moves of the game, plays an important role. The domain theory does not help in forming strong cases, although it might, but in producing cases that can be manipulated by the genetic portion of the system to form strong cases.

To describe how the system produces cases we must first talk about the general operation of it. The game begins with the human making a move. The system must now produce its own move. If the system can find a case in memory that contains, in its frames, the exact game history[1], the next move in the case is returned as the system's move. However, if no such case can be found, the last white move and a legal move returned by the domain theory form a new case that is added to the case library, and the legal move is returned as the system's move. The new, two move case that was formed by the system can also be extended in the following manner. If it, or any other case, was used to return the last black move, but does not have a move in response to the next white move because the case is at its last frame, and no other cases in memory have a move in response to white's move, rather than using the last white move and a legal move returned by the domain theory to form a new case, these moves are added to the end of the last used case, and the legal move returned as the system's move. The effect of this is that rather than forming another smaller case, the case that was used in the last turn is extended by two more moves.

The fact that the domain theory returns any legal move, not the best, leads to the fact that the strategies formed from the domain theory are not necessarily good; however, the system must detect this. A case library of strong strategies is the system objective.

4.2 Reinforcement Schema: Modified Bucket Brigade

The Bucket Brigade Algorithm [Goldberg, 1989; Westerdale, 1989; Grefenstette, 1988] is an apportionment of credit algorithm for classifiers[2]. We propose a modified and renamed algorithm for the apportionment of credit of cases called the Reinforcement Schema for Cases (RSC). Whether for a classifier or for a case the requirement is similar; genetic operators require a method of ranking cases or classifiers. The rank, a case's strength or fitness value, is a measure of the performance of the case in its environment.

A schematic of the RSC processes is shown in figure 4.1. Initially, all cases matching the current environmental features are located. The cases are said to be activated for duty (ready to return a solution). To perform a case's duty, an auction is held to determine the case that has the right to return its solution. When only one case is returned, it automatically wins the auction and returns its solution. When there is more than one case in the active case set, each case makes a bid for the right to return its solution. The bid of a case is proportional to its strength. Stronger cases make larger bids than weaker ones. For example, a simple bid function could be the average strength of a case's frames:

$$\text{Case Bid} = \frac{\sum_{i=1}^{\text{Size(Case)}} \text{FrameStrength}(i)}{\text{Size(Case)}}$$

1 The game history is the previous sequence of game moves to a selected depth.

2 A classifier is a special form of chromosome string that encodes a rule for use in a classifier system.

The case with the largest bid wins the right to perform its duty and the other cases are deactivated.

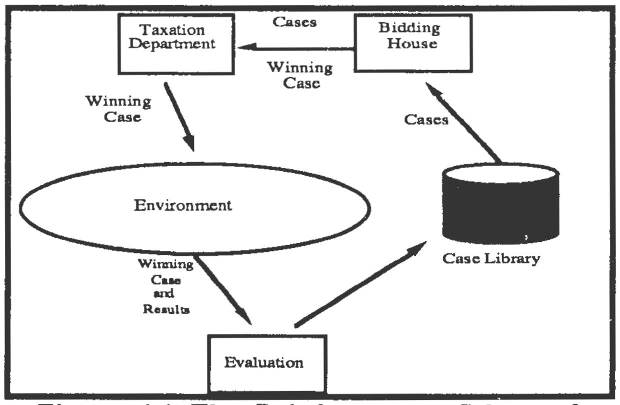


Figure 4.1 The Reinforcement Schema for Cases

All cases that bid for the right to perform their duty must now pay for that right. Each is taxed by the amount of its bid. The effect is that each case has a decrease in its strength. How does a case increase its strength? This depends on how well the winning case performs its duty in the environment. If a case performs adequately, its strength regains the amount of its bid; its strength is now at least as strong as it was before it performed its duty. If the case performed exceptionally well, its bid and a reward amount are added to its strength, making it a little stronger than is was before performing its duty. Finally, if the case performed poorly, the bid is not returned and a penalty amount is subtracted from its strength. Cases that perform well will have their strength increased, while cases that appear to be strong but perform poorly, will have their strength decreased more than the cases that lost the right to perform their duty. This gives these secondary cases a better chance at performing their duty on the next iteration, given the same conditions.

4.3 Approach: Simple Classifier System For Cases

Our approach called the Simple Classifier System for Cases (SCSC), and its three components: CBR, genetic algorithms, and reinforcement schema, are shown in figure 4.2 in a closed learning loop.

We describe the architecture by way of a checker example. As a starting point for the SCSC, we begin with the case library. The case library contains a collection, possibly empty, of source cases. In the checker example, each case represents a checker strategy. For this example, we start with an empty case library, although initially we could seed the library with hand-coded cases. As long as there is a limited domain theory, and an initial learning period, the SCSC will develop its own cases.

Next, the environment produces an input problem for which an action or response is required. The input situation is used as a probe and all appropriate cases that have a chance of providing a solution to the input problem are retrieved from the case library. This is the normal process of CBR retrieval. These activated cases are then passed to the *Bidding House*. Here the cases bid for the right to return their responses. As a result of the auction, the case with the largest bid is found.

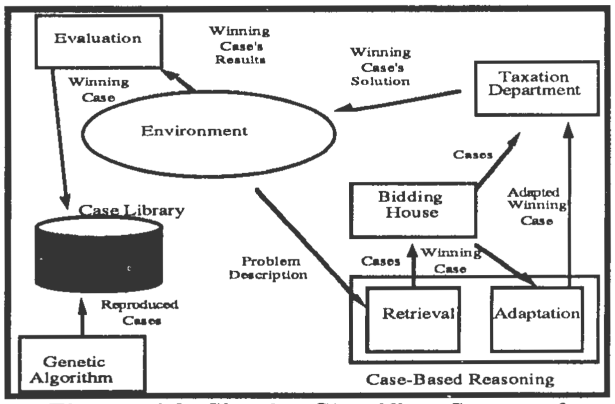


Figure 4.2 Simple Classifier System for Cases

In the checker example, the bid of a case is a function of the case's strength which is a function of the individual strengths of the case's frames. The bid is the normalized ratio of the number of frames that match the problem description in the retrieval process divided by the number of frames in the case. The function causes cases that have a greater number of matching frames to be favored over smaller cases that match the same number of frames.

At this point, a limited domain theory may be required. It is limited because it provides a minimal amount of information to enable the formation of initial cases. For example, if there are no cases retrieved by the CBR retrieval process, the domain theory can form a short simple case and return it as the winning case. In the checker example, this would be equivalent to starting a case with the last move of the opponent followed by any legal system move. It is not difficult for a domain theory to compute a legal move as long as it knows the rules of the game. What the domain theory is missing is knowledge as to what constitutes a good move. This, as we will see, is where the genetic algorithm and the RSC form an important part of the approach.

The winning case is now passed to the CBR adaptation process so final modifications can be made, ensuring that its response fits the input problem. This is a link back to the CBR process. Here we do not present any such adaptations, but if so desired, further adaptations can be made using traditional CBR adaptation techniques. The adapted winning case and all other activated cases are passed to the *Taxation Department* for the decrease of their strengths by their bids.

The winning case's response is now passed to the environment for use. All other activated cases have no chance of getting their strengths back, except on further iterations of the learning loop. It is only the winning case that has a chance of gaining its strength back and more. This all depends on how well the case performs in the environment. The goal is to select the best case for use in the environment. The environment provides feedback as to how well, or poorly, the case performs, in the form of three results: -1, 0, and 1. The feedback is used by the *Evaluation* component of the approach. In the checker example the environmental evaluation is as follows:

```
If the move produces a king
      or jumps an opponent Then
            a 1 is returned
Else If the move produced an opponent jump Then
            a -1 is returned
Else
            a 0 is returned
```

The *Evaluation* component takes the following actions based on these three results:

```
Case Evaluation of
0 :    add the case's bid to its strength;
1 :    add the case's bid and a reward value to its
       strength;
otherwise: {the evaluation is a -1}
       penalize the case's strength further by a
       penalty amount
```

In the checker example, a frame's maximum allowable strength is 2, and the reward and penalty values 1. Cases initially have all their frames' strengths set to 0.2. After completing the evaluation, the modified cases are updated in the case library.

The processes of case retrieval, adaptation, and evaluation are repeated for a fixed[3] number of iterations, after which the CRAC is performed on the case library, removing the bad cases, keeping the good ones, and adding new ones based on variations of existing strong cases. The result is that over time a library of strong cases will be generated from a initial, possibly empty, set of original cases.

The SCSC and the RSC provide systematic methods for obtaining strong cases; genetic algorithms provide a novel method of case adaptation, but still rely heavily on the past history of the case so as to not alter the case too drastically; and CBR is a method of reasoning from cases. The combination of the three leads to a complete approach to learning and reasoning in a dynamic environment. Each of the above components aids the others in performing their task.

5 Results

The results described in this section are produced from a human playing a 4x4 checker game against a SCSC as described in the last section. The goals of the SCSC are to produce new, strong cases using genetic operators, and to have the overall strength of the case library improve with each successive generation. Using a 4x4 checker game integrated with the approach, we demonstrate that these goals are met.

The following results show how the SCSC operates on a volatile parameter set. With these parameter settings, genetic operators are applied with a probability of 1. That is, given the correct conditions, the genetic operator will be performed. Also, the number of cases in the library is small compared to the number of possible cases in the domain. This means that there is a high turn-over of cases from generation to generation. The end result is that new cases are added and removed at a high rate.

Before describing the individual results, we describe the common factors affecting the SCSC. Initially the case library is empty, no seed cases are provided, and the only domain theory is the simple rules of checkers; that is when called upon the domain theory will return a random legal move for black, the system player. This is an important point. Good case information is not added to the case library to begin with. Therefore, its initial strength is zero. Just relying on the domain theory to build our cases would result in cases built at random - since the domain theory only returns random legal moves. If the moves returned were poor, the cases would be constantly loosing strength, and with such cases the library would experience an overall decrease in strength, not an increase as desired. Each session consists of forty checker games where the human and system player alternate moves under the rules of checkers. The CRAC is performed every *n* games, where *n* is set as a parameter.

The results plot the average case strength and case library strength against the number of games played using the following volatile parameter set.

Maximum Case Library Size = 10
Genetic Algorithm Performed Every 2 Games
Probability of Mutation, Crossover, Division, and
Connection = 1

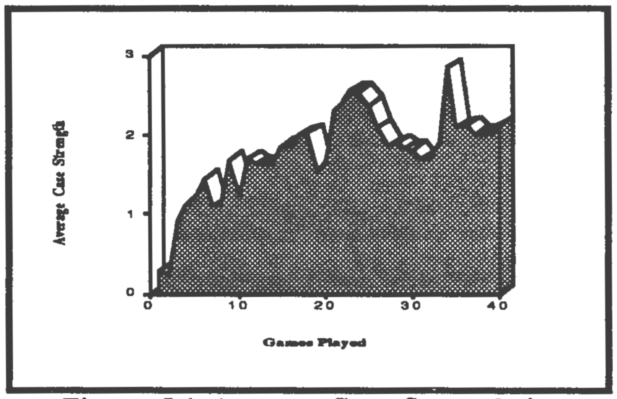


Figure 5.1 Average Case Strength in a Volatile Environment

In figure 5.1, the average case strength is plotted against the number of games played. As shown in the graph, there is a steep rise in the average strength of a case over the first few games because there are no cases in the library to begin with. However, after the case library is filled up with the domain theory cases, it is easy to see that the overall average case strength continues to increase. There are many local minima and maxima, but this is the nature of using genetic operators. There are major advancements in strength followed by retreats; however, the number of advancements is larger than the number of retreats. From

[3] System parameter set upon initialization of the SCSC.

the graph in figure 5.1 it is easy to see that the overall strength of the cases is increasing with every game played.

Figure 5.2 plots the strength of the entire case library against the number of games played. The result is the same. After a fast jump in the total strength of the library, as result of the addition of new cases to the case library by the domain theory, the overall strength continues to increase as a result of the application of the genetic algorithm and the RSC. Other experiments using different parameter settings experienced the same results - an overall increase in the strength of the cases in the case library

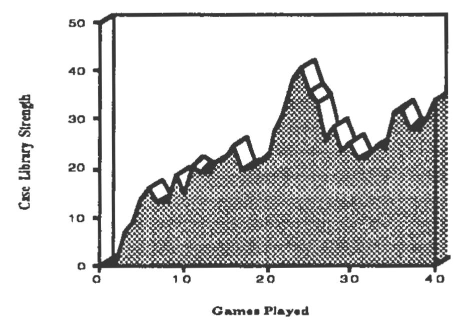


Figure 5.2 Case Library Strength in a Volatile Environment

6 Summary

CBR is a powerful model of reasoning based on the human ability of reminding. Although powerful, still many issues remain to be solved to make CBR a more practical and useable method of reasoning. Our approach has been to take other AI methods and integrate them, where appropriate, to extend CBR by providing solutions to some of the open issues. In particular, we have adapted genetic techniques for use by CBR to aid in forgetting and novel case construction. Adaptation is in fact the central theme of CBR; adapt solutions when appropriate to provide, in this case, novel methods of extending CBR.

We feel that our approach could benefit anybody concerned with the growing size of case memory over time and with issues of the timely detection of redundant and invalid cases. However, the full advantages of the approach only accrue to systems for which a 'fitness' function can be devised to help eliminate bad new cases.

The use of genetic techniques has proven quite helpful in the formation of a self-perpetuating learning model. Although the genetic case structure is constructed as a general structure, not connected to any specific domain, the application of the model to another domain would help to further develop the characteristics that cases must have for the further use of genetic techniques. The current strength of a case, for instance, is shown as a function of the strengths of individual frames (slots, game moves, etc) of a given case. If cases do not have a frame structure, what other methods of assessing case strength are there? Different methods of calculating strength and fitness have been proposed by genetic algorithm researchers and would be of great interest to those wishing to alter the method of credit apportionment for cases.

7 References

[Antonisse, 1987] Antonisse, H. J., and Keller, K. S., Genetic Operators for High-Level Knowledge Representation, Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum, 69-76, 1987.

[Bickel, 1987] Bickel, A. S., Bicker, R. W., Tree Structured Rules in Genetic Algorithms, Representation, Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum, 77-81, 1987.

[Bradtke, 1988] Bradtke, S. & Lehnert, W. G., Some Experiments with Case-based Search, Proceedings: Case-Based Reasoning Workshop, Morgan Kaufmann, 80-93,1988.

[Deugo, 1989a] Deugo, D. L., and Oppacher, F., Case-Based Techniques for Learning and Repairing Plans, Proceedings of The Second International Symposium on Artificial Intelligence, Monterrey, N.L. Mexico, 1989.

[Deugo, 1989b] Deugo, D. L., Extending Case-Based Reasoning Using Rule-Based and Genetic Techniques, Master's Thesis, Carleton University, School of Computer Science, 1989.

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

[Grefenstette, 1988] Grefenstette, J. J., Credit Assignment in Genetic Learning Systems, Proceedings of the Seventh National Conference on Artificial Intelligence,Morgan Kaufmann Publishers, 596-600, 1988.

[Kolodner, 1987] Kolodner, J.L., Simpson, K., and Sycara-Cryanski, K., A Process Model of Case-Based Reasoning in Problem Solving, Proceedings of the Sixth National Conference of Artificial Intelligence, Morgan Kaufmann, 284-290, 1987.

[Kolodner, 1988] Kolodner, J. L., Extending Problem Solver Capabilities Through Case-Base Inference, Proceedings: Case-Based Reasoning Workshop, Morgan Kaufmann, 21-30, 1988.

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

[Sycara, 1988] Sycara, K., Using Case-Based Reasoning for Plan Adaptation and Repair, Proceedings: Case-

Based Reasoning Workshop, Morgan Kaufmann, 425-434, 1988.

[Westerdale, 1989] Westerdale, T. H., A Defense of the Bucket Brigade, Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, 282-290, 1989.

Building System Specifications Using Explanation-Based Learning with an Incomplete Theory

Jean Genest[†]
Département de Mathématiques
Collège Militaire Royal de St-Jean
St-Jean-sur-Richelieu, Québec,
J0J 1R0,Canada
jean@uotcsi2.bitnet

Stan Matwin
Department of Computer Science
University of Ottawa, Ottawa, Ontario,
K1N 6N5 Canada
stan@uotcsi2.bitnet

Abstract

Explanation-Based Learning (EBL) is an analytical learning method where a concept is learned by building an explanation of a training example. One weakness of EBL is its inability to explain when the theory is incomplete. This paper presents a systematic approach to deal with imperfect theories based on abductive reasoning, analogical reasoning and case-based reasoning. The approach is presented in the context of the system LISE (Learning in Software Engineering). LISE converts a user requirement for a software module into an operational module definition, using a (possibly incomplete) specification of this and other modules as the domain theory. When the training example can be explained in this theory, the result of EBL is a specification for the user requirement given in terms of primitive operations. If the training example is unexplainable, LISE will extend the incomplete specification to cover the new user requirement.

1 Introduction

In the software development life cycle, one of the early phases consists of transforming informal user requirements into a more formal software specification. This phase, called the analysis phase, relies mainly on the experience and creativity of the system analyst. Consequently, different specifications can be produced for the same requirement and, even worse, the resulting specification can suffer from incompleteness and inconsistency.

A software system usually implements one or more activities normally carried out by some people. The approach used by system analysts to produce the specification of a software system should consist in specifying externally observable activities [Yau et al. 1986]. The specifications of the activities are integrated together resulting in the specification of the entire software system.

To build the specification of an activity, the analyst takes an example and extracts from it a generalized sequence of actions. Actions in the sequence are subsequently refined until they can be implemented. Suppose that banking is our area of application. The following scenario shows an activity called `withdraw` as observed by an analyst:

```
Bob walks up to the teller and indicates
that he wants to withdraw $100. The teller
verifies if Bob has an account. The teller
then ensures that the balance of Bob's
account - $150- is greater that $100. The
teller will then subtract $100 from the
balance of Bob's account and will give the
money to Bob. Bob walks away.
```

The analyst will identify all the relevant actions in the activity in their correct order. The analyst will discard the actions which, according to his understanding of the domain, are irrelevant, e.g. `Bob walks up to the teller` and `Bob walks away`. The analyst will also generalize the actions so that they do not only apply to constants introduced by the specific example. The predicate logic rendering of the specification produced by the analyst is :

```
withdraw(Person,Amount) <-
   account(Person,Account),
   balance(Account,Balance),
   Balance > Amount,
   sub_fm_balance(Balance,Amount),
   issue_money(Person,Amount).
```

This paper presents a system designed to implement the process followed by the analyst in the scenario above.The system called LISE (Learning In Software Engineering) transforms activities representing user requirements into corresponding formal specifications. The formal specification for each activity is produced in a notation combining frames and predicate logic. LISE uses EBL

[†] This work was done at University of Ottawa

(Explanation-Based Learning) to build explanations of the activities using a domain theory. The domain theory is the specification of software system being developed. It includes specifications of all the activities known to this point. The result of EBL is the specification for the user requirement in terms of primitive operations. The primitive operations do not need any further specification: they may represent library modules or abstract data type operations which are already available.

If the user requirement is an activity for which no specification exists in the domain theory, regular EBL will not be able to build the explanation. This is called the incomplete domain theory problem, and is typical of any practical application of EBL. LISE will extend the domain theory using abduction, analogical reasoning and case-based reasoning.

2 Explanation-based Learning

Explanation-Based Learning (EBL) is an analytical learning method where a definition of a concept from some domain is learned using a single *training example* [DeJong et al. 1986],[Mitchell et al. 1986]. The process of EBL uses a *domain theory* , i.e. a set of rules pertinent to the domain of the goal concept. The domain theory is used to construct an explanation, or a deductive proof, of how the training example is an instance of the *goal concept*. The goal concept is often expressed in terms which are not useful for the particular application of the concept definition. The explanation creates a new definition of the goal concept, expressed in terms which are operational, i.e. adequate for a given use of the concept. A concept definition is considered operational when it is expressed in terms which satisfy a stated *operationality criterion*.

An explanation in EBL is build using rules. The antecedents of the rules are satisfied using facts from the training example or using the consequents of other rules. A common assumption in EBL is that the domain theory contains all the rules required to build the explanation of each positive training example. Unfortunately, real domain theories tend to be incomplete [Ellman 1989], [Rajamoney et al. 1987]. Instead of producing a full explanation as described above, the incomplete domain theories will produce one or more partial explanations. A partial explanation is an explanation containing proven and unproven antecedents. Antecedents are proven using training example facts, or using the consequent of a rule for which all antecedents were proven. The unproven antecedents are antecedents for which no facts were found in the example, and no rules could be used to prove the antecedents.

In LISE, if the training example is a new user requirement, the domain theory will miss some of the rules necessary to build an explanation. LISE will produce one or more partial explanations. LISE will then attempt to build a plausible explanation using abduction, analogical reasoning and case-based reasoning.

[Ellman 1989] mentions that the effort in EBL research addresses the problems of justified generalization, chunking, operationalization and justified analogy. With regard to that classification, our work addresses the problem of operationalization since our objective is to translate a non-operational expression (i.e. user requirement) into an operational one (i.e. a specification).

3 The domain theory in LISE

In LISE, the domain theory represents the specification of a software system. In the particular software system considered here, the domain theory will be the specification of a banking system.

The specification of the system is given in terms of objects and operations applicable to objects. The objects represent structural properties of the system. The operations represent behavioral properties of the system.

The domain theory consists of frames arranged in a hierarchy and allowing multiple inheritance of properties. Each frame specifies an object or an operation using a set of properties. One common property of all frames is `isa` which links a frame to its parents. Frames representing objects are located under the top-level frame called `ENTITY`. Similarly, the frames representing operations are located under the frames `ACTION` and `TRANSACTION`. Operations specified as `ACTION`s in the hierarchy are non-primitive operations. Non-primitive operations are defined by a precondition stating the condition under which a primitive operation can be applied and a procedure which includes the single primitive operation. Primitive operations are atomic and they are not defined anywhere. Operations specified as `TRANSACTION`s in the hierarchy are defined using primitive and non-primitive operations arranged in a fixed sequence. Figure 1 illustrates four frames taken from a domain theory containing the specification of a banking system. Each frame of the domain theory is transformed into a rule prior to executing EBL.

During the analysis phase, the analyst has to take into account the objectives of the organization for which the system is being designed. The objectives of the organization are specified by specific goals. An example of a goal for an organization such as a bank is protect_bank_interest. The analyst will have to keep this goal in mind when writing the specification of any banking transaction. Goals are included in the domain theory of LISE. A goal is attached to a property of an entity or to an operation which is intended to enforce it. An example of a goal in the domain theory is

```
goal(credit_margin(Person,Margin),
     protect_bank_interest)
```

```
withdraw(Person,Amount)
isa:  TRANSACTION
  precondition:
    account(Person,Account)
  procedure:
    debit(Account,Amount)
    issue_money(Person,Amount)

deposit(Person,Amount)
isa:  TRANSACTION
  precondition:
    account(Person,Account),
  procedure:
    issue_money(Person,Amount)
    credit(Account,Amount)

debit(Account,Amount)
isa:  ACTION
  precondition:
    balance(Account,Balance)
    Balance > Amount
  procedure:
    sub_fm_balance(Account,Amount)

credit(Account,Amount)
isa:  ACTION
  precondition: nil
  procedure:
    add_to_balance(Account,Amount)
```

Figure 1. Frames of the domain theory for banking

which means that the goal `protect_bank_interest` is enforced by the property `credit_margin` of `Person`. Goals are essential to the analogical reasoning process employed to extend the domain theory.

4 LISE in a nutshell

LISE is an integrated learning system consisting of three modules.

a. Explanation-based learning for LISE (ELLI). ELLI is the first module receiving the training example. The main engine of ELLI is the conventional EBL process in which the deductive inference found in regular EBL was enhanced by the addition of abductive inference.

b. LISE's Analogical ReaSoner (LARS). LARS is a module designed to build a new specification from a set of partial explanations. LARS will build a plausible explanation by re-using the proven antecedents of the partial explanations and by applying analogical reasoning to replace the unproven antecedents. The plausible explanation will be used to create the new frames which will be added to the domain theory.

c. CAse-based Reasoning for LISE (CARL). If ELLI and LARS fail to provide the specification for a training example, the module CARL will use its case-base in order to build the new specification.

4.1 Explanation-based learning for LISE (ELLI).

ELLI receives the training example representing a user requirement and attempts to explain it according to the domain theory. If an explanation is produced, then all is well. The result will be the specification for the training example given in terms of primitive actions. Figure 2 shows such a scenario where the training example is the user requirement for the transaction `withdraw(bob,100)`. The training example is given as a sequence of four features. The result produced by ELLI is the specification for the transaction `withdraw(Person,Amount)` which is given in terms of five features which are more general than features of the training example. The new feature is the constraint `Balance > Amount` which is part of the specification and was satisfied by the training example values for `Balance` and `Amount`. The feature `address(bob,101_Colonel_by)` is considered irrelevant and was not integrated in the specification of `withdraw`.

```
The training example is: withdraw(bob,100)
The facts are:
   account(bob,acc_1)
   address(bob,101_Colonel_by)
   balance(acc_1,150)
   sub_fm_balance(acc_1,100)
   issue_money(bob,100)

Result:
  withdraw(Person,Amount)
Specification:
  account(Person,Account),
  balance(Account,Balance),
  Balance > Amount,
  sub_fm_balance(Account,Amount),
  issue_money(Person,Amount)
```

Figure 2. A complete explanation scenario

The specification from the domain theory that were used in the construction of the specification of figure 2 are `withdraw(Person,Amount)` and `debit(Account,Amount)` as shown in figure 1.

When a explanation is not possible using the deductive inference, ELLI applies abduction [Pople 1973]. Abduction is the generation of hypotheses, which, if true, would explain observed facts. More precisely, if the rule `Q <- P` and the fact `Q` are given, then the desired abductive conclusion would be `P`. `P` can be characterized as being a hypothesis because there could exist another rule `Q <- P'` which would have been used to derive `Q`.

Abduction is used to cope with partial explanations. Using the training example features as a set of facts, ELLI attempts to draw hypotheses from the unproven antecedents using abduction. If hypotheses can be drawn for each unproven antecedent, the partial explanation can be completed. Using the above example of the rule `Q <- P` and `Q`, `P` would be an unproven antecedent and `Q` would be a fact given as a training example feature. If `P` is the only unproven antecedent, a hypothesis can be drawn to account for the occurrence of `Q` in the training example, and the partial explanation can be completed.

The next scenario illustrates the usage of abduction to complete a partial explanation. The training example is `withdraw(bob,100)` of figure 2 where the fact `account(bob,acc_1)` was replaced by `client(bob)` (See figure 3).

```
The training example is: withdraw(bob,100)
The facts are:
   client(bob)
   address(bob,101_Colonel_by)
   balance(acc_1,150)
   sub_fm_balance(acc_1,100)
   issue_money(bob,100)
```

Figure 3. The training example on which ELLI uses abduction

The partial explanation produced for the training example contains the unproven antecedent `account(bob,Account)` as shown in the explanation tree of figure 4. The partial explanation generated can be changed into a plausible explanation by using abduction to transform `account(bob,Account)` as a hypothesis for the training example feature `client(bob)`. The domain theory rule `client(Person) <- account(Person,Account)` was used to perform abduction. That rule is the part of the definition of the entity `client`.

Again, we see that some irrelevant features of training example, `address(bob,101_Colonel_By)` and `phone(bob,992-2318)`, were not used in the explanation.

4.2 LISE's analogical reasoner (LARS)

When no complete explanation is possible using the domain theory, ELLI will generate one or more partial explanation. A partial explanation is an explanation containing some proven and some unproven antecedents. Each partial explanation is provided by the specification of a transaction from the domain theory which contains one or more facts belonging to the training example.

Initially, LARS will rank the partial explanations according to a score attributed to each partial explanation based on its coverage of the training example. The heuristic proposed to score the explanations is:

a. reward a partial explanation for each common feature it shares with the training example,

b. penalize a partial explanation for each of its unproven leafs,

c. penalize a partial explanation for each feature of the training example that was left unaccounted for, and,

d. slightly penalize a partial explanation for each abductive inference that was used in its construction.

LARS will attempt to build a plausible explanation by first re-using the proven antecedents of the best partial explanation. Unproven antecedents of the best partial explanation are replaced by *analogous* training example facts, i.e. by those facts that share the same goal with the

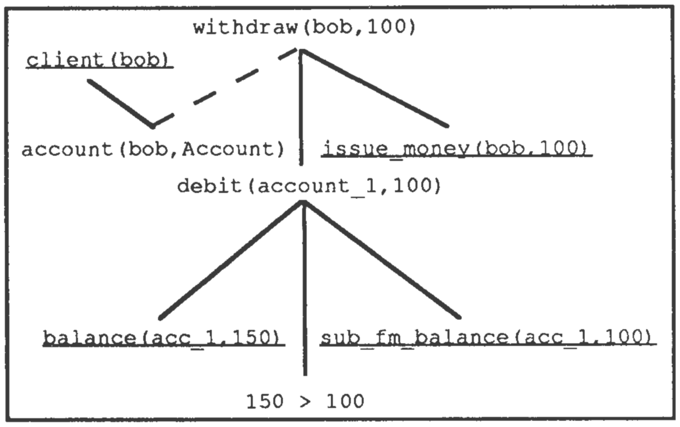


Figure 4. EBL with abduction

unproven antecedents. The plausible explanation so obtained will be used to augment the domain theory so that it includes the specification for the new transaction. Since the theory is augmented during learning, LARS performs a form of constructive learning [Muggleton et al. 1988].

LARS will be demonstrated using the training example for `borrow(bob,1000)`. The training example is shown in figure 5.

The domain theory is incomplete because the specification of the transaction `borrow` required to explain the training example is missing. Consequently, two partial explanations will be produced. Figure 6a and figure 6b illustrate partial explanations that were obtained using the transactions `withdraw` and `deposit`. The dashed lines in the partial explanations indicate which antecedents were left unproven. The underlined antecedents in the partial explanations indicate which antecedents were found as training example features. The ranking heuristic determined that the best partial explanation was the one in figure 6a obtained using `withdraw(Person,Amount)`.

```
The training example is: borrow(bob,1000).

The facts are:
  account(bob,acc_1)
  credit_margin(bob,3500)
  record_loan(bob,1000)
  issue_money(bob,1000)
  car(bob,car_of_bob)
  value(car_of_bob,12000)
```

Figure 5. The training example for borrow(Person,Amount)

Figure 7 shows that the proven antecedents of withdraw are re-used in the plausible explanation of `borrow`. The unproven antecedents were replaced by selected analogous training example features. A training example feature is selected to replace an unproven antecedent if it shares the same goal. In the `borrow` example, `account(bob,acc_1)`, `issue_money(bob,1000)` and the operator ">" were all re-used. The antecedent `balance(Account,Balance)` was replaced by the training example feature `credit_margin(bob,3500)` because they have the same goal: `protect_bank_interest` Similarly, `sub_fm_balance(Balance, Amount)` was replaced by `record_loan(bob, 1000)` because they share the same goal: `record_transaction`. LISE asked the user for a name to replace `debit(Account, Amount)` in the plausible explanation. The user provided the name `grant_loan`.

Training example features that were not re-used nor selected are deemed irrelevant. In the example, `car(bob,car_of_bob)` and `value(car_of_bob,12000)` are irrelevant.

Initially, partial explanations were obtained because the domain theory did not contain the frames for `borrow` and for `grant_loan`. To circumvent the incompleteness problem, a plausible explanation was build from the partial explanation obtained using `withdraw`. The next step is to synthesize the missing frames from the plausible explanation and to insert

them in the domain theory. The frames that produced the partial explanation are used as a guide to create the new frames from the plausible explanation. The structure of the frame of `withdraw` is used to create the frame of `borrow` and the structure of the frame of `debit` is used to build the frame of `grant_loan`. Figure 8 shows the frames added to the domain theory.

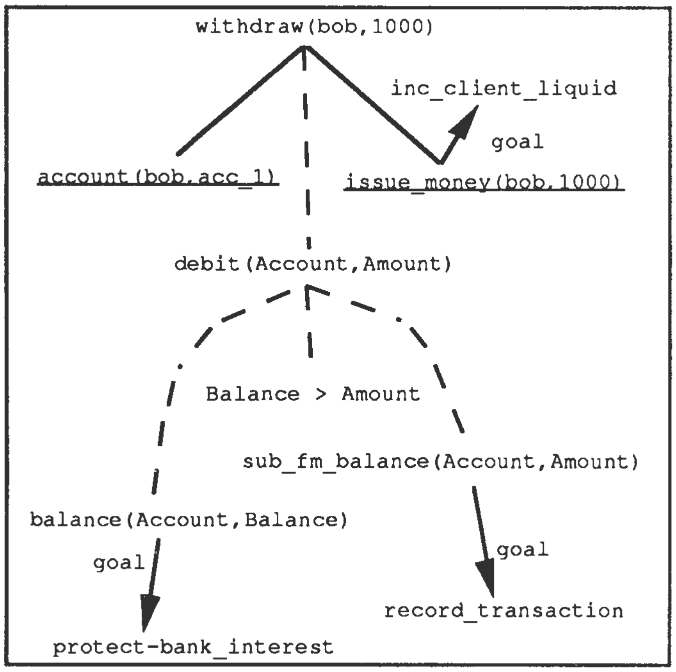


Figure 6a. Partial explanation for borrow produced using withdraw

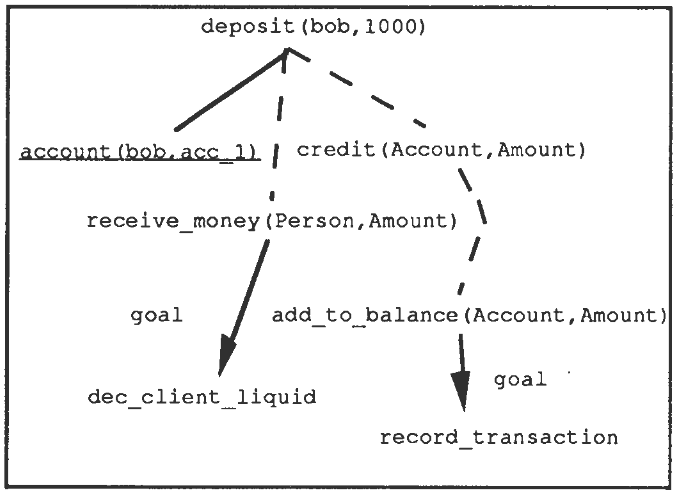


Figure 6b. Partial explanation for borrow produced using deposit

A single partial explanation might contribute to explain several features of the training example while leaving out other relevant ones. Such a situation can be suspected when several partial explanations match different features of the training example. Figure 9 pictures the training example `transfer(Person,Amount)` which will be learned using two partial explanations.

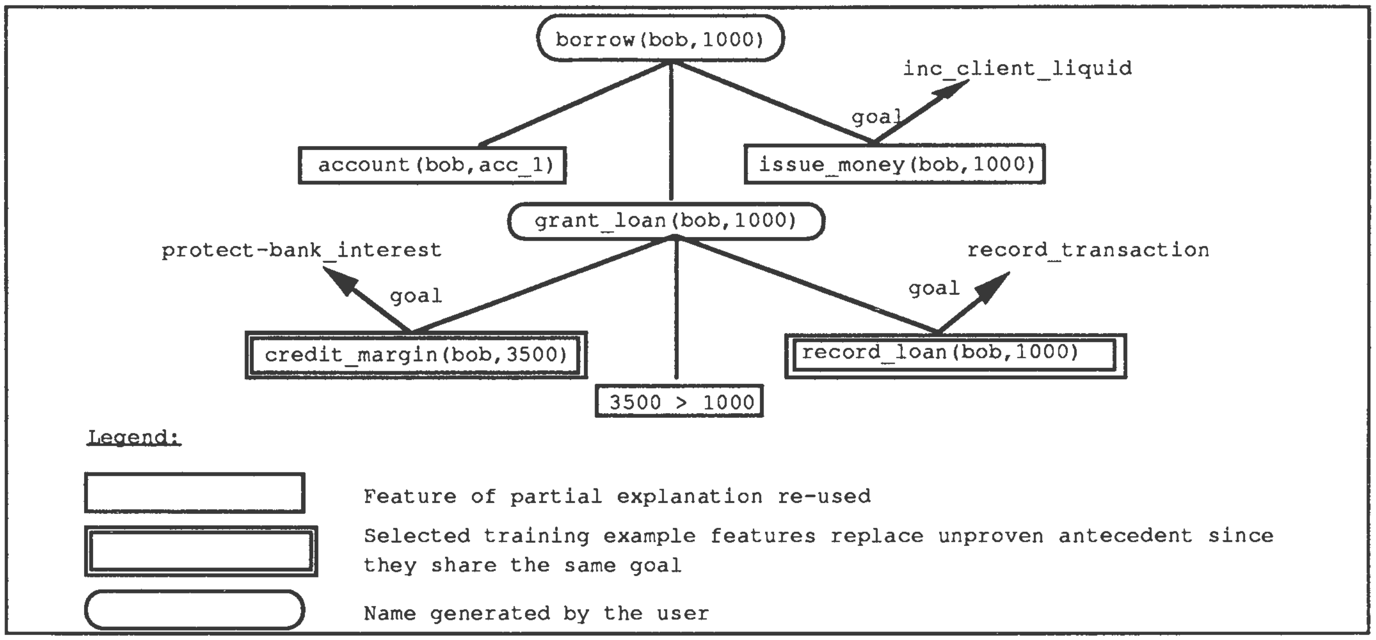


Figure 7. The plausible explanation for borrow produced using withdraw

```
Name: grant_loan(Person,Amount)
 isa: ACTION
   precondition:
     credit_margin(Person,Margin)
     Margin > Amount
   procedure:
     record_loan(Person,Amount)

Name: borrow(Person,Amount)
 isa: TRANSACTION
   precondition:
     account(Person,Account)
   procedure:
     grant_loan(Person,Amount)
     issue_money(Person,Amount)
```

Figure 8. The frames for grant_loan and for borrow

```
The training example is: transfer(bob,100).

The facts are:
   account(bob,acc_1)
   account(bob,acc_2)
   phone(bob,992-2318)
   balance(acc_1,350)
   sub_fm_balance(acc_1,100)
   add_to_balance(acc_2,100)
```

Figure 9. The training example for the transaction transfer(Person,Amount).

The domain theory is incomplete since it does not include the frame of `transfer`. Consequently, partial explanations are produced (Figure 10a and 10b). It is interesting to note that the partial explanation obtained using `withdraw` covers some training example features (the underlined antecedents) while the one obtained using `deposit` covers other features. In that case, we recognize that a single partial explanation will not provide enough foundation to build the plausible explanation. Both partial explanations will be integrated to yield the plausible explanation of `transfer` (Figure 11). The unproven antecedents `issue_money` of `withdraw`, and `receive_money` of `deposit`, do not raise any problem since the goal hierarchy indicates that they can be mutually removed when the `person` and the `amount` are the same.

After the plausible explanation was build, the frame (Figure 12) for `transfer` is synthesized and integrated with the domain theory. It is the only frame added to the domain theory since the frames of `debit` and `credit` already exists.

4.3 Case-based reasoning for LISE (CARL)

LARS processes the training examples using a rule-based approach. This approach proved to be very effective in providing an explanation and in extending the domain theory when required. However, LARS can run into difficulty for some training examples when a lack of background knowledge hinders the usage of analogical reasoning. CARL is a case-based reasoning system designed to retrieve previous cases to apply to these training examples.

The cases in our system represent transactions which are not explained with the domain theory because of their complexity or because they are exceptions to general rules. For example, suppose that according to the domain theory, a check may be cashed by a person only if that person owns an account. Suppose that there is a single

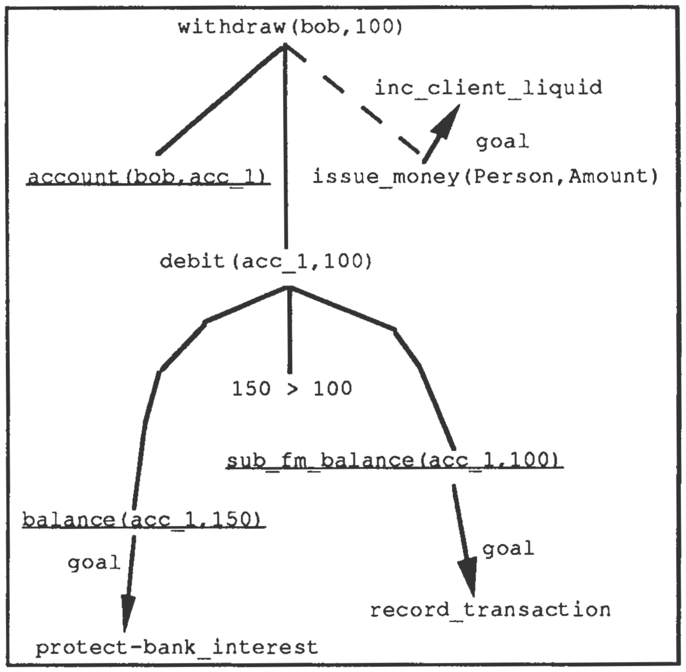


Figure 10a. Partial explanation for transfer produced using withdraw

exception to this rule for travellers checks. Instead of refining the domain theory to cover the exception, a case will be inserted in the case-base so that, when the domain theory fails to explain a training example where a person cashes a travellers check, a case will be applied by CARL.

The transactions in the case-base are composed of a precondition and a procedure containing properties of entities and operations. We call these properties and operations case features. CARL will retrieve a case for a training example if there is a match between the case features and the training example features and if the order of the case features is preserved in the training example. Detail of the matching process are described in [Genest90]. When more than one case is retrieved, a heuristic similar to the one used to rank the partial explanations will select the most applicable one.

The cost of matching and the likelihood that a case be applicable make the case-based approach of CARL secondary to the rule-based approach of LARS.

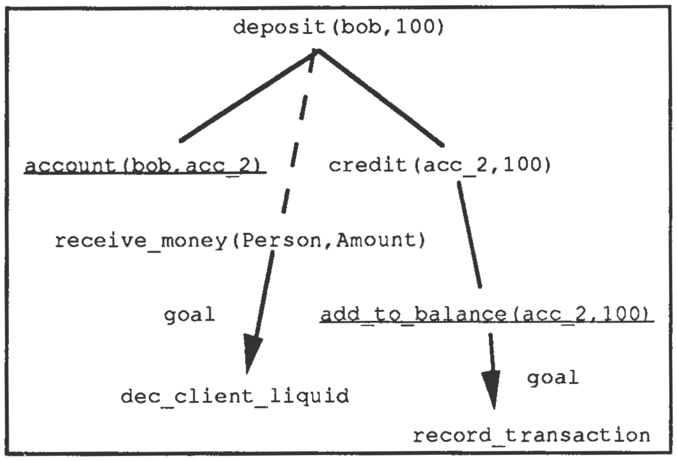


Figure 10b. Partial explanation for transfer produced using deposit

5 Conclusion

This paper has presented a learning method in which EBL is used in concert with an incomplete domain theory. The approach to deal with the incomplete theory is by integrating abduction, analogical reasoning and case-based reasoning. Inasmuch as our system augments the deductive closure of its domain theory by adding rules into it, it achieves knowledge-level learning. This is seldom the case for EBL systems.

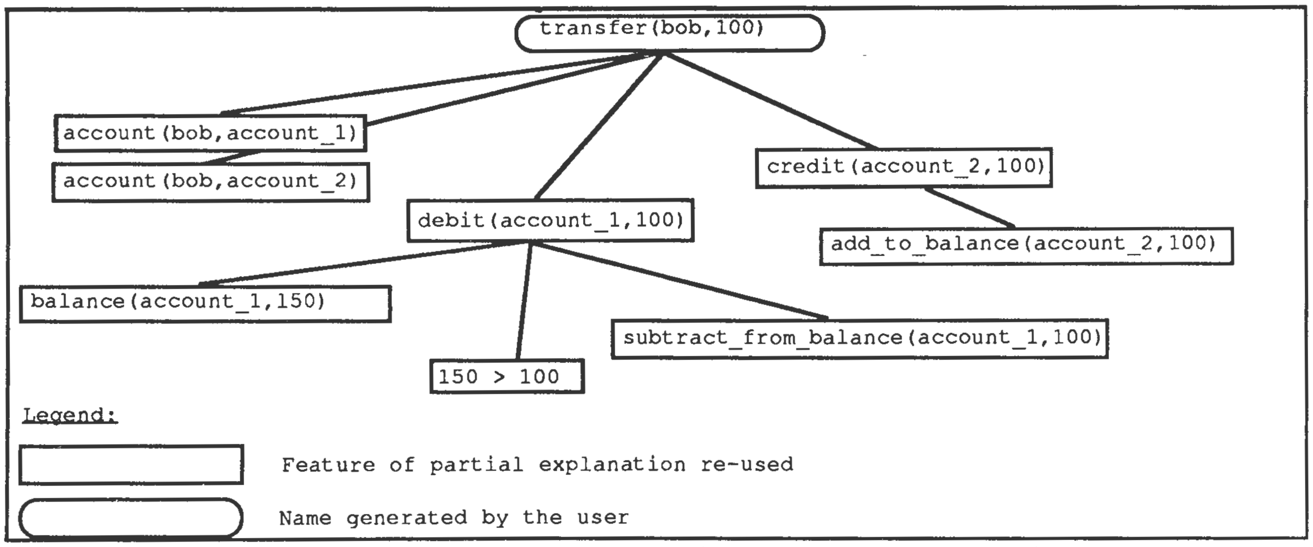


Figure 11. The plausible explanation for transfer produced using withdraw (the goals are omitted for clarity)

```
Name: transfer(Person,Amount)
 Parents: [transaction(Person)]
   precondition:
     account(Person,Account1)
     account(Person,Account2)
   procedure:
     debit(Account1,Amount)
     credit(Account2,Amount)
```

Figure 12. The new frame for transfer added to the domain theory.

The cost involved in LISE to extend the incomplete domain theory does not increase considerably beyond that of normal EBL. If the case-based approach needs to be used for a training example, the cost will increase more but will still be reasonable.

[Ellman 1989] divides the methods to handle the incomplete domain theories into the analytical methods and the empirical methods. The usage of abduction, analogical reasoning and case-based reasoning categorizes LISE as an analytical methods. Other analytical methods require that a pair of training examples with similar functions be presented simultaneously [Hall 1988] or they need an experimentation theory to refine the domain theory [Rajamoney 1988]. Empirical methods deal with incomplete domain theories [Pazzani 1988] [Fawcett 1989] by conjecturing rules to fill holes in the partial explanation and, using subsequent training examples, empirically refine the conjectured rules.

LISE was successfully implemented in Prolog. A specification for a small banking system was developed. We are currently working on the specification of a fleet management system.

6 Future research

As in any non-empirical learning system, the quality of the specification learned by LISE is limited by the amount of knowledge initially contained in the domain theory and in the case-base. A future goal is to make LISE less dependent on the initial knowledge.

A future plan for LISE is to produce executable specifications. This would require that the primitive actions of the specification be mapped to an implementation such as a database management system or a library of Ada packages. Similarly, the static part of the specification would have to be mapped to a data structure. LISE could then execute the specification for given user requirements providing instant design capability.

Acknowledgement

The research described here was done at the Knowledge Acquisition Laboratory in the Department of Computer Science, University of Ottawa. The work of the Laboratory is sponsored by Natural Sciences and Engineering Research Council of Canada, the Ontario University Research Incentive Fund, Cognos, Inc., and the University of Ottawa. Financial support for the first author was provided by the Department of National Defence.

We would like to thank Boris Plante for a useful review of an early draft of this paper.

References

[Dejong et al. 1986] DeJong, G. F. and Mooney, R.. "Explanation-Based Learning: An Alternative View", *Machine Learning*, vol. 1, pp. 145-176, 1986.

[Ellman 1989] Ellman, T. "Explanation-Based Learning: A survey of Programs and Perspective.", *ACM Computing Surveys*, vol. 21, no. 2, pp163-221, 1989.

[Fawcett 1989] Fawcett, T. (1989) "Learning from Plausible Explanations", Procs. of *9th International Conference on Machine Learning*, Ithaca, pp. 37-39, 1989.

[Genest90] Genest, J. "Building Software Specifications using Explanation-based Learning with Incomplete Theories", Master's Thesis, University of Ottawa, March 1990.

[Hall 1988] Hall, R. J. "Learning By Failing to Explain: Using Partial Explanations to Learn in Incomplete or Intractable Domains", *Machine Learning*, vol. 3, no. 1, pp. 45-77, 1988.

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

[Muggleton et al. 1988] Muggleton, S. and Buntine, W. "Machine Invention of First-Order Predicates by Inverting Resolution", Procs. of *Fifth Machine Learning Conference*, pp. 339-352, 1988.

[Pazzani 1988] Pazzani, M. "Integrated Learning with Incorrect and Incomplete Theories", Procs. of *5th National Conference on Artificial Intelligence*, pp. 555-559, 1988.

[Pople 1973] Pople, H. E., Jr. "On the Mechanization of Abductive Logic", Procs. of *3rd IJCAI*, 1973.

[Rajamoney et al. 1987] Rajamoney, S. and DeJong, G. "The Classification, Detection and Handling of Imperfect Theory Problem", Procs. of *10th IJCAI*, Milan, Italy, 1987.

[Rajamoney 1988] Rajamoney, S. A. "Experimentation-Based Theory Revision", Procs. of *AAAI Spring Symposium on Explanation-Based Learning*, pp. 7-11, 1988.

[Yau et al. 1986] Yau, S. S. and Tsai, J. J.-P. "A Survey of Software Design Techniques", *IEEE Transactions On Software Engineering*, vol. 12, no. 6, pp. 713-721, 1986.

Using Distribution-Free Learning Theory To Analyze Chunking

William W. Cohen
Rutgers Computer Science Department
New Brunswick, NJ 08903
U.S.A.

Abstract

In the last few years, a theoretical framework, variously called *probably approximately correct (PAC) learning theory* and *distribution-free learning theory*, has been developed for analyzing the behavior of a broad class of learning algorithms. This paper extends this framework to the problem of improving a program's performance, with the aim of developing a methodology for the analysis of common program improvement learning techniques such as macro-operators [Fikes *et al.*, 1972], EBL [Mitchell *et al.*, 1986], and chunking [Laird *et al.*, 1986]. The framework is then used to evaluate a simplified chunking mechanism for a class of heuristic search programs. Several of the predictions made by the model have been confirmed by recently published experiments [Mooney, 1989; Tambe and Rosenbloom, 1989].

1 Introduction

Research in machine learning has been primarily focused on two problems. The first problem, called *symbol-level learning (SLL)*, is to improve the performance of some sort of program given examples of its behavior on typical inputs. The second problem, called *knowledge-level learning (KLL)*, is to identify one particular concept from a list of hypotheses, given examples of members and non-members of the concept.[1]

In the last few years, a theoretical framework, variously called *probably approximately correct (PAC) learning theory* and *distribution-free learning theory*, has been developed for analyzing the behavior of a broad class of knowledge-level learning algorithms [Blumer *et al.*, 1986; Valiant, 1984]. This paper extends this framework to the problem of symbol-level learning, or improving a program's performance. The goal of this paper is to develop a methodology for the analysis of common program improvement learning techniques such as macro-operators [Fikes *et al.*, 1972], EBL [Mitchell *et al.*, 1986], and chunking [Laird *et al.*, 1986].

[1]This distinction, and the terminology used here to describe it, was introduced in [Dietterich, 1986].

The paper begins by defining a notion of "performance improvement"; our definitions parallel Valiant's definitions of PAC-learnability [Valiant, 1984]. We then describe a framework for analysis of learning programs, and use the framework to evaluate a chunking mechanism for a particular class of search programs. Several of the predictions made by our model have been confirmed by recently published experiments [Mooney, 1989; Tambe and Rosenbloom, 1989].

Because of space limitations, proofs have been either sketched or omitted. Interested readers are referred to [Cohen, 1989] for details.

2 The goal of learning

The goal of SLL is to improve a program. In general, there are two ways in which a program p may be improved: the quality of the solutions which p generates may be improved, relative to some metric on solution quality, or the run-time of p may be improved. In this paper we consider only the problem of improving the run-time of programs; we use the term *performance improvement learning (PIL)* to describe this task. A PIL program, then, will keep solution quality constant and improve performance. We also restrict our attention to improvement of programs which use search to find solutions to problems given as input.

The definitions below make precise the sort of performance improvement considered acceptable. In the definitions, $TIME(p,s)$ is some measure of the time required by the program p to process the input s, $SIZE(p)$ is some measure of program size, and D is a probability distribution on initial problem states. D will sometimes be referred to as the *run-time environment* of the program p. Also, the *average case time complexity* of a program p on a set S, denoted $AVGTIME_D(p,S)$, is defined as

$$\sum_{s \in S} D(s)TIME(p,s)$$

Definition 1 (ϵ-approximation) *A search program q is called an* ϵ-approximation of p *(with respect to D) if, for an s drawn randomly according to D,*

$$Prob(q(s) \neq p(s)) < \epsilon$$

A search program q is called an improving ϵ-approximation of p *if it is an ϵ-approximation of p and $AVGTIME_D(q,S) < AVGTIME_D(p,S)$.*

The goal of PIL is to reliably produce an improving ϵ-approximation using a bounded number of training examples and computational resources.

The fact that an improvement program q need not be correct for *all* possible inputs, but only for *most* inputs, is important. Unless p is poorly coded, improving its performance by purely automatic means is a very difficult problem. However, it is often the case that the inputs to p will follow some pattern which is unknown to the implementor of p. In this case, it is reasonable to try to infer this pattern of inputs and then optimize the program p for those inputs. Since the inference about the environment is not likely to be perfect, it is likely that the optimized program will fail for some small fraction ϵ of the inputs.

Notice that the definition does not specify what the improvement program q will do on the remaining ϵ of the inputs. Two reasonable choices would be for q to simply abort, or for q to call the original program p.

3 A framework for analysis of PIL programs

One of the reasons for the success of the Valiant framework for learnability is that there are simple procedures for establishing the success of a learning algorithm within the framework. One of such procedure rests on the following basic result. Define a *sample of p drawn according to D* to be a set of pairs $\{\langle s_1, p(s_1)\rangle, \ldots, \langle s_m, p(s_m)\rangle\}$ such that each s_i is drawn according to the probability distribution D. We will consider learning programs that dynamically build up a sample via calls to an oracle for p. If H is a *hypothesis space*, a set of functions which are possible conjectures of a learning program $LEARN$, let $[H]_n$ denote $\{q \in H : SIZE(q) < n\}$, and define the *dimension* of $[H]_n$ (written $\dim([H]_n)$) to be $\log_2 |[H]_n|$. In [Natarajan, 1989] the following result is given:[2]

Theorem 1 (Natarajan 89) *Let* $\dim([H]_n)$ *be polynomial in n, and let LEARN always request a sample S and return a hypothesis* $q \in H$ *such that*

1. $p(s) = q(s)$ *for every problem* $s \in S$,
2. *there is no* $q' \in H$ *which meets condition (1) such that* $SIZE(q') < SIZE(q)$.

Then if S exceeds a certain size (which is polynomial in $1/\epsilon$, $1/\delta$, *and the size of p), LEARN outputs an* ϵ-*approximation of p with probability at least* $(1-\delta)$.

A similar result can be developed for the PIL framework. Define q to be *optimal on S with respect to H and D* if

$$\forall q' \in H,\ AVGTIME_D(q, S) \leq AVGTIME_D(q', S)$$

Also let $DOMAIN(p)$ denote the set of problems for which p is capable of finding solutions. The following theorem can now be stated.

Theorem 2 *Let* $\dim([H_p]_n)$ *be polynomial in n, and let LEARN(p) always request a sample S return a program* $q \in H_p$ *such that*

- $p(s) = q(s)$ *for every problem* $s \in S$,
- *there is no* $q' \in H$ *which meets condition (1) such that* $SIZE(q') < SIZE(q)$,
- *q is optimal on DOMAIN(q) with respect to* H_p.

Also let q_{opt} *be the program which is optimal on DOMAIN(p) with respect to* H_p *and D, and assume that there is some* $q_{exact} \in H_p$ *which is identical in behavior and performance (but not necessarily in size) to p.*

Then if p is suboptimal *on DOMAIN(p) with respect to* H_p, *and if S exceeds a certain size (which is polynomial in* $1/\epsilon$, $1/\delta$, *and the size of* q_{opt}*), LEARN outputs an* improving *ϵ-approximation of p with probability at least* $(1-\epsilon)(1-\delta)$.

Proof: Natarajan's result tells us that q will agree with p on all but ϵ of the inputs with probability at least $(1-\delta)$; the remainder follows from theorem 7 of [Cohen, 1989]. ■

There are three changes in this theorem, relative to the Natarajan result. First, the program p is now known, and is assumed to be suboptimal. Second, the hypothesis space H_p is derived from p (typically, via a set of "improvement operations" such as adding control rules, etc) rather than being independent of p. Finally, learning program is allowed to take time proportional, not to the program p, but to the unknown optimal program in H_p.

This theorem, like the previous one, describes a methodology for evaluating learning algorithms. The theorem says that in order to achieve performance improvement, it is sufficient to find a program q which agrees with p on a number of inputs, and which is optimal in both size and in average-case performance. Requiring optimality in *size* ensures that the learner converges to a q which is behaviorially a good approximation of p. Requiring optimality in *performance* ensures that q's average case performance is better.

It is not typically the case that optimality in size will correspond to optimality in performance; this may occur, however, if every $q \in H_p$ is a set of decision rules. In this case, it may be that a smaller set of rules will always take less time to use than a larger set of rules.

4 Analysis of a Chunking Mechanism

The learning algorithm described in this section is an idealized version of chunking: it is similar in flavor to the techniques described in [Fikes *et al.*, 1972; Laird *et al.*, 1986; Mitchell *et al.*, 1986]. We first discuss the various components of the learning mechanism: the class of programs which can be improved, the time measure relative to which programs are improved, the hypothesis space H_p, and finally the learning algorithm.

4.1 The Class of Improvable Programs

We assume that the program which is to be improved is a heuristic search program using the A search algorithm as presented in [Nilsson, 1987]. A program can be thought of as a triple $p = (h, O, F)$ where h is the *heuristic function*, O is the set of available *operations*, and F is the set of *solution states*. The output of a heuristic search

[2]This result generalizes a result from [Blumer *et al.*, 1986] to functions.

program given an initial problem state s is defined to be the sequence of operations which leads to a final state.

Following Nilsson [Nilsson, 1987], a search program is called *monotone* if, $\forall s_i, s_j,\ h(s_i) \leq h(s_j) + c(s_i, s_j)$, where $c(s_i, s_j)$ denotes the cost of the operator o such that $o(s_i) = s_j$. Henceforth we assume that all programs are monotone.

The time measure, denoted $NODES(p, s)$, is the number of nodes expanded by p in finding a solution to the problem with initial state n. This has been chosen because it is closely related to actual CPU time (as shown in [Cohen, 1989]), but is also analytically tractable. $AVGNODES_D(p, S)$ denotes the average-case time complexity with respect to this measure.

We assume the operations to be deterministic, in the sense that applying an operator to a state produces a unique result; it is not necessary, however, to assume that there is a finite set of such operators. It is also not necessary for our analysis to assume that each operator produces a single subgoal: in other words, we also consider *decomposable production systems* [Nilsson, 1987] (sometimes called problem reduction problem solvers) which are guided by a monotone heuristic function. A *decomposable production system* is a problem-solving system which decomposes an initial problem into a set of independent subproblems, and then recursively solves the subproblems. The recursion ends when all the subproblems have been "solved" by reducing them to a set of *final states*. For a decomposable production system, the output of the program is an not an operator sequence but an *operator tree*, which specifies the partial ordering of problem decomposition operators used to reach a solution state. An operator tree will be represented as a tree with operator labels on the arcs; the nodes of the tree, which are unlabeled, correspond to problem states.

A more detailed description of this extension to the heuristic search can be found in [Cohen, 1989].

4.2 The Hypothesis Space

For state-space search problem solvers, a concise representation of a set of operator sequences is a rooted, directed graph in in which every path through the graph is an operator sequence. We will call such a graph a *sequence graph*. The natural extension for decomposable production systems is a *sequence hypergraph*[3], which is a concise representation of a set of operator trees.

Let p be a search program. We define $RESTRICT(p, G)$ to be a search program q which contains only operators from p to which certain preconditions have been added: preconditions which constrain q to apply operators in an order consistent with some path in the sequence (hyper)graph G. We also restrict q to use the same rule as p uses to decide which of several nodes of equal value to expand next: for instance, if p choses between nodes of equal heuristic worth by always expanding the most recently generated node first, then q must also operate in this manner.

The set $H_p = \{q : \exists G\ q = RESTRICT(p, G)\}$ will be used as our hypothesis space. Our size measure $SIZE(q)$ on programs in this space is just the number of maximal-length paths in the hypergraph G which begin at the root.

The correspondence between a program $q \in H_p$ and a set of chunks is close, but not obvious. Note that applying a single chunk is equivalent to executing a series of primitive operators, so there is a close correspondence between a single chunk and an operator tree. A sequence hypergraph is essentially a collection of operator trees in which the heuristic function used by the original search program has been retained, and is used to decide in what order to apply operator trees. Another advantage of the hypergraph representation is that operator sequence hypergraphs provide a simple means of combining common prefixes of two operator trees, which eliminates a possible source of inefficiency for the problem solver. These optimizations are necessary to ensure that chunking does not actually *degrade* performance.

Another important difference is that in most chunking mechanisms, when it is known that a specific series of operators *Seq* will be applied, some sort of partial evaluation[4] is performed on *Tree* so that the series of operators can be applied more efficiently. We have elected to ignore this aspect of chunking, and emphasize the role of chunking in reducing search. Our results indicate that even without partial evaluation, chunking can provably improve performance.

4.3 A Chunking Mechanism

We are now ready to present a learning algorithm which makes use of sequence hypergraphs to construct an improvement of a program p. The program CHUNK — code for which is shown in Figure 1 — constructs a sequence hypergraph HTREE which contains exactly those operator sequences which have been used to solve the example problems. These sequences are organized, as the name suggests, into a hypertree structure, in which the longest common prefixes of two trees will be shared. This eliminates a possible source of inefficiency. The variable n counts the number of times that CHUNK rejects a hypothesis, and is used to determine when to stop extending the sample.

To illustrate the operation of CHUNK, consider the simple decomposable production system shown in figure 2. This program uses a series of rewrite rules to verify that containers are cups. For example, the initial problem TESTCUP(CONT1) could be re-written as the set of subproblems

$$\{\text{TESTSTABLE(CONT1)}, \text{TESTLIFTABLE(CONT1)}\}$$

and, after additional rewrites, eventually solved.

Consider now using CHUNK to improve this program. Assume that the first problem/solution pair returned by $SOLUTION_p$ is for the problem TESTCUP(CONT1). The operator tree for the solution to this problem is

[3]A hypergraph is also sometimes called an AND/OR graph. Our usage of this term follows [Nilsson, 1987].

[4]By *partial evaluation* we refer to some sort of program transformation which is meaning-preserving, but which improves efficiency. An example of partial evaluation is the goal regression done in EBG [Mitchell *et al.*, 1986].

```
Algorithm CHUNK($\epsilon$, $\delta$, $p$):
inputs: real numbers $\epsilon$, $\delta$; a program $p$
outputs: a program $q$ which approximates $p$

$m \leftarrow n \leftarrow 0$
$HTREE \leftarrow empty\_htree()$
while $m \leq confirm(\epsilon, \delta, n)$ do
    $m \leftarrow m + 1$
    $(x, Tree) \leftarrow SOLUTION_p()$
    if $Tree \notin HTREE$ then
        $htree\_extend($
          $root(HTREE), Tree)$
        $m \leftarrow 0$
        $n \leftarrow n + 1$
    endif
endwhile
return $RESTRICT(p, HTREE)$
```

```
Subfunction $confirm(\epsilon, \delta, n)$ :
return $\frac{1}{\epsilon} \ln \frac{2n^2}{\delta}$

Subroutine $htree\_extend(v, Tree)$ :
inputs: a vertex $v \in HTREE$; a tree $Tree$
modifies: the hypertree $HTREE$

$r \leftarrow$ the root of $Tree$
$o_1 \leftarrow$ the operator labeling $r$
if $\exists$ a hyperedge $(v, \langle w_1, \ldots, w_k \rangle)$
  labeled $o_1$ then
    for $1 = 1, \ldots, k$ do
        $T_i \leftarrow$ the $i$-th subtree of $r$
        $htree\_extend(w_i, T_i)$
    endfor
else
    replace $v$ with a copy of $Tree$
endif
```

Figure 1: The Performance Improvement Learner CHUNK

Operators:

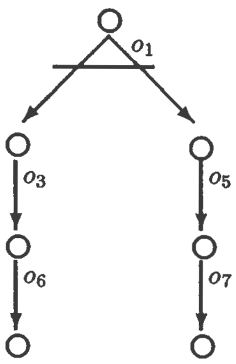
o_1: TESTCUP(x) $\rightarrow$ TESTSTABLE(x), TESTLIFTABLE(x)
o_2: TESTCUP(x) $\rightarrow$ TESTDISPOSABLE(x), TESTLIFTABLE(x)
o_3: TESTSTABLE(x) $\rightarrow$ TESTFLATBOTTOM(x)
o_4: TESTLIFTABLE(x) $\rightarrow$ TESTCONIC(x)
o_5: TESTLIFTABLE(x) $\rightarrow$ TESTHANDLE(X)
o_6: TESTFLATBOTTOM(CONT1) $\rightarrow$ T
o_7: TESTHANDLE(CONT1) $\rightarrow$ T
o_8: TESTCONIC(CONT2) $\rightarrow$ T
o_9: TESTFLATBOTTOM(CONT2) $\rightarrow$ T

Solution States: { T }

Heuristic function:

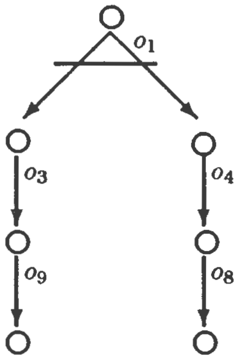
h(decomposition tree) = number of non-T leaves

Figure 2: An example search program p

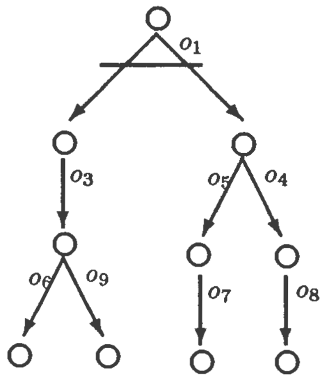


Since this tree is not contained in the (empty) hypertree HTREE, the routine *htree_extend* is called. The effect is add to HTREE a copy of this tree, and to designate the root of the tree as the root of HTREE. So HTREE is now a copy of the tree above. Additionally, m is reset to zero, and n is incremented to 1.

Now suppose that the next problem/solution pair provided
by $SOLUTION_p$ is the problem TESTCUP(CONT2), which has the solution below:



Again, this operator tree is not contained in HTREE, to *htree_extend* is called. The result is to modify HTREE, producing the following hypergraph.



CHUNK will continue to incrementally add operator trees to this hypergraph until the hypergraph remains unchanged after $confirm(\epsilon, \delta, n)$ consecutive examples. At this point, CHUNK will assume that the hypergraph is sufficent, and return a program q which has the same operator set and heuristic function as the initial search program, but which uses this hypergraph to constrain its search.

There are several difference between CHUNK and conventional chunking mechanisms. One difference between CHUNK and conventional chunking mechanisms is that most chunking mechanisms do not learn solutions to all problems, but only for a carefully selected set of problems or subproblems on which chunking is likely to help — for instance, problems on which the default search strategy does poorly, or commonly occuring subproblems. Construction of such "problem filters" is an active area of research [Laird *et al.*, 1986; Minton, 1988]. CHUNK contains no such filtering mechanism, although one could be easily added by filtering the output of the oracle $SOLUTION_p()$.

Another difference is that CHUNK does not generalize solution sequences in any way. This deficiency is addressed in section 5.

4.4 Analysis of CHUNK

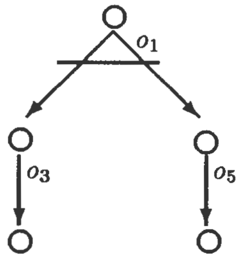
Theorem 2 can now be used to establish the following result; see the proof of theorem 5 in [Cohen, 1989] for details.

Theorem 3 *Let q_{opt} be the program which is optimal on $DOMAIN(p)$ with respect to H_p and D. Then if p is suboptimal on $DOMAIN(p)$ with respect to H_p, CHUNK outputs an improving ϵ-approximation of p with probability at least $(1-\epsilon)(1-\delta)$, making at most $\frac{n}{\epsilon}\ln\frac{2n^2}{\delta}$ calls on $SOLUTION_p()$. Further, CHUNK runs in time polynomial in the size of its inputs and the values returned by $SOLUTION_p()$.*

5 Abstracting Operator Sequences

A common extension to chunking is abstraction of the chunks which are learned, so that they are more broadly applicable. Consider a variant of CHUNK called CHUNK-A in which, rather than adding a only single operator tree when HTREE proves to be inadequate, a large number of trees "similar" to the new solution tree *Tree* are added. One possible implementation of CHUNK-A is to store in HTREE some sort of an "abstract description" of *Tree*. Since this abstract description will also match many other sequences, the effect is as if many trees similar to *Tree* had been stored.

The most common context in which operator sequences are abstracted is in decomposible production systems. Recall that the solution "sequences" of a decomposible production system are naturally represented as trees. These trees can be abstracted by pruning away leaf nodes. A natural way of restricting a search program by an such an abstract operator tree is to require existing links in the tree to be followed, but to allow any sequence of applicable operators in solving subproblems which have been pruned. For example, one way to abstracting the operator sequence for TESTCUP(CONT1) in the example is to simply discard the final arcs in the tree.



In EBL, this technique is called *operationality pruning* [Mitchell *et al.*, 1986]; a similar technique for planning problems is called *plan abstraction* [Sacerdoti, 1974]. I will refer to a version of CHUNK which has been modified in this way as CHUNK-A; see [Cohen, 1989] for details.

The primary advantage of using abstraction is that the convergence rate of the learning algorithm can be improved. Suppose, for instance, that an abstraction function is used which maps 100 different operator sequences to a single abstract description. CHUNK-A can learn these 100 different operator sequences from a single example, whereas CHUNK may require as many as 100 examples. Of course, if only one of the 100 different operator sequences which are mapped to the same abstract description is actually used, then CHUNK-A will learn at precisely the same rate as CHUNK. Formalizing this argument leads to the following theorem.

Theorem 4 *Let CHUNK-A use an abstraction function which maps A operator descriptions to B abstract descriptions, let $\alpha = A - B$, and consider a learning problem for which CHUNK requires m examples. Then, if n is the size of the program output by CHUNK, and if CHUNK-A receives exactly the same sequence of examples as CHUNK:*

1. *CHUNK-A will require* at most *m examples.*
2. *CHUNK-A will require* at least *$m - \frac{\alpha-1}{\epsilon}\ln\frac{2n^2}{\delta}$ examples.*

The price that is paid for possibly more rapid convergence is that the program q produced by CHUNK-A is less restricted, in the sense that the hypergraph used to restrict p contains more paths, and thus potentially not as efficient. There is, however, one interesting case in which there is no efficiency loss.

Theorem 5 *Let S be a class of problem states such that the heuristic function h is perfect on S: that is, no nodes not directly on the path to the solution found are visited in solving a problem $s \in S$. Let CHUNK-A use an abstraction function which maps every possible solution sequence for a problem $s \in S$ to the same abstract description, let q_1 be the program learned by CHUNK using a fixed set of examples, and let q_2 be the program learned by CHUNK-A using the same set of examples. Then*

$$\forall s : q_1(s) = q_2(s), \quad NODES(q_2, s) = NODES(q_1, s)$$

In other words, abstraction carries no efficiency penalty if the abstraction function classifies as "plan details" or "operational subgoals" only those subproblems which can be solved *without search.* A corollary of this result is that CHUNK-A, with such an abstraction function, has the same computational properties as CHUNK.

6 Related Work

Since these results were developed, several experimental results have been published which support the predictions made by the model. One of the conditions necessary to guarantee the optimality of a set of chunks is that the heuristic function be monotonic. In [Mooney, 1989] results are given which show that unrestricted use of EBL in a depth-first search problem solver leads to performance degradation, whereas use of use of EBL in a breadth-first search problem solver leads to performance improvement; note that breadth-first search uses a monotonic search function, and depth-first search does not. A second conditions necessary to guarantee performance improvement in our model is that abstraction must be constrained to only "abstract away" subproblems which can be solved without using search. In [Tambe and Rosenbloom, 1989], results are given which show that using an arbitrary abstraction function on chunks in SOAR can cause performance degradation. This degradation can be eliminated by constraining the abstraction function to produce chunks which can be matched in linear time: that is, without search. The argument made in [Tambe and Rosenbloom, 1989] to support the experimental claim relies on the condition-sharing done by the RETE algorithm, an operation similar in effect to the operator tree merging done in CHUNK.

In [Greiner and Likuski, 1989] a formal analysis of EBL is made, under a different assumption. Greiner assumes that cached solution paths are incorporated as additional operations, augmenting the set of primitive queries. A method for incorporating redundant rules and constructing an optimal ordering of rules is described. However, the optimality is with respect to a single query, not a set of queries; the general problem of finding an optimal search strategy in an arbitrary redundant search space is shown to be NP-hard. In our research, in contrast, we assume the original set of primitive operators will be discarded, and a set of cached solution paths used in its stead. In doing this, we sacrifice completeness of the optimized problem solver, but make tractable the process of finding an optimal search strategy. The negative results of [Greiner and Likuski, 1989] can be viewed as support for our contention that it is necessary to trade off some coverage to tractably obtain performance improvement on the remaining inputs.

In [Mahadevan *et al.*, 1988; Natajaran and Tadepalli, 1988] is an alternative formalization of the problem of improving the performance of search programs. They take the goal of learning to be improvement in the asymptotic time complexity of problem solving; the principle means considered for doing this is inductively learning the conditions under which operators are useful to apply. Two practical difficulties with applying their framework are, first, that verifying the preconditions of their theorems may be difficult in a poorly understood domain (this has been verified for one formalization of the problem of symbolic integration, however [Natajaran and Tadepalli, 1988]), and second, that implementation of the oracles needed by the learning algorithm appears to require breadth-first search: in contrast, implementation of the $SOLUTION_p()$ oracle requires only heuristic search.

7 Conclusion

The ultimate goal of the research described in this paper is to develop a sound and workable methodology for the analysis of common program improvement learning techniques such as macro-operators [Fikes *et al.*, 1972], EBL [Mitchell *et al.*, 1986], and chunking [Laird *et al.*, 1986]. To this end, we have extended the distribution-free learning framework introduced by Valiant [Valiant, 1984]; we defined a notion of program improvement which, paralleling Valiant's definition, allows the improving program to *approximate* the original program, and then extended the basic result of the Valiant learning framework to the problem of performance improvement learning. An alternative theoretical analysis of performance improvement learning [Greiner and Likuski, 1989] suggests that this assumption may be necessary for tractable learning.

We then use this learnability framework to evaluate a chunking mechanism for a particular class of search programs. Several of the predictions made by our model have been confirmed by recently published experiments; in particular, the requirements that the original search program use a monotonic heuristic function, that common conditions be merged, and that abstraction be greatly constrained are supported by experimental evidence in [Mooney, 1989; Tambe and Rosenbloom, 1989].

Acknowledgements

I would like to thank Alex Borgida, Jack Mostow, Chun Liew and Ravi Boppana for useful discussions in the early stages of this research, and for comments on an early draft. The paper also benefitted from the comments of two anonymous referees. The author was supported by a Marion Johnson Fellowship during the initial stages of this research, and later by an AT&T Bell Laboratories Fellowship.

References

[Blumer *et al.*, 1986] Anselm Blumer, Andrej Ehrenfeucht, David Haussler, and Manfred Warmuth. Classifying learnable concepts with the Vapnik-Chervonenkis dimension. In *18th Annual Annual Symposium on the Theory of Computing.* ACM Press, 1986.

[Cohen, 1989] William W. Cohen. Solution path caching mechanisms which provably improve performance. Technical Report DCS-TR-254, Rutgers University, 1989.

[Dietterich, 1986] Thomas Dietterich. Learning at the knowledge level. *Machine Learning Journal*, 1, 1986.

[Fikes *et al.*, 1972] Richard Fikes, Peter Hart, and Nils Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[Greiner and Likuski, 1989] Russell Greiner and Joseph Likuski. Incorporating redundant learned rules: A preliminary formal analysis of EBL. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.* Morgan Kaufmann, 1989.

[Laird *et al.*, 1986] John Laird, Paul Rosenbloom, and Allen Newell. Chunking in SOAR: The anatomy of a general learning mechanism. *Journal of Machine Learning*, 1, 1986.

[Mahadevan *et al.*, 1988] S. Mahadevan, B. K. Natajaran, and P. Tadepalli. A framework for learning as improving problem-solving performance. In *Proceedings of the 1988 Spring Symposium on EBL.* AAAI, 1988.

[Minton, 1988] Steven Minton. Quantitative results concerning the utility of EBL. In *Proceedings of the Seventh National Conference on Artificial Intelligence.* AAAI, 1988.

[Mitchell *et al.*, 1986] T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1), 1986.

[Mooney, 1989] R. J. Mooney. The effect of rule use on the utility of explanation based learning. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.* Morgan Kaufmann, 1989.

[Natajaran and Tadepalli, 1988] B. K. Natajaran and P. Tadepalli. Two new frameworks for learning. In *Proceedings of the Fifth International Conference on Machine Learning.* Morgan Kaufmann, 1988.

[Natarajan, 1989] B. K. Natarajan. On learning sets and functions. *Machine Learning Journal*, 1(4), 1989.

[Nilsson, 1987] Nils Nilsson. *Principles of Artificial Intelligence.* Morgan Kaufmann, 1987.

[Sacerdoti, 1974] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5(2):115–135, 1974.

[Tambe and Rosenbloom, 1989] Milinde Tambe and Paul Rosenbloom. Eliminating expensive chunks by restricting expressiveness. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.* Morgan Kaufmann, 1989.

[Valiant, 1984] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11), November 1984.

A Semantic Approach to Analogical Reasoning*

William T. Hunt
Mary McLeish
Department of Computing and Information Science
University of Guelph
Guelph, Ontario, Canada N1G 2W1
william@snowhite.cis.uoguelph.ca
mary@snowhite.cis.uoguelph.ca

Abstract

If analogical reasoning programs are going to either have a broad application or approach the capacity of ordinary human thinking, then the mechanisms must become more complex or the semantics must become richer or both. Little research is being done to discover what a broad collection of semantics can contribute to general purpose analogical reasoning.

This paper presents a methodology for determining which meaningful concepts to use in general analogical reasoning. It uses a broad range of metaphors and incorporates human subject understanding. The reasoning mechanism operates on a broad semantic base using a thesaurus hierarchy for interactions between the target and source components of metaphors.

1 Introduction

Studies of metaphor and analogical reasoning have been explored for a long time by many disciplines[Shibles, 1971; von Noppen et al., 1985; Helman, 1988] with much of the current research coming from artificial intelligence [Prieditis, 1988; Eskridge, 1989; Hall, 1986; Hall, 1989; Leishman, 1989; Wolstencroft, 1989]. This paper describes a semantic approach to analogical reasoning.

1.1 Scope, Diversity, and Human Understanding

1.1.1 Scope

To approach ordinary human thinking, general analogical reasoning programs need either more complex mechanisms or much richer semantics, or both. With the notable exception of the CYC project[Lenat et al., 1986], little research is being done with a broad collection of semantics for general purpose analogical reasoning. The methodology discussed herein addresses this point by using a hierarchy containing a **wide scope of concepts** (called schemata) taken from **Roget's Thesaurus** [1987] spanning the two classes of abstract relations and space.

*This work was supported by NSERC grant # 99185220 and NSERC grant # A4515.

1.1.2 Diversity

AI approaches tend to be based on a small collection of well-used examples. Both the program and the representation of a particular analogy can appear to be built for each other. **More diverse metaphors** need to be handled for a program to work beyond a narrow range of domains. The methodology presented here uses a collection of 260 wide-ranging nonliterary metaphors[Katz et al., 1988] and takes precautions to independently represent each of the primary (target and source) concepts (without knowledge of the metaphor).

1.1.3 Human Understanding

Current approaches underplay the human understanding of metaphor. While psychological studies are incorporated in AI research, more understanding of human metaphorical reasoning is required. The methodology below applies **human understanding** to machine-generated meanings in determining the concepts used in metaphorical reasoning.

This paper presents a novel methodology for determining which semantics are useful in general analogical reasoning. It uses a broad range of metaphors and incorporates human subject understanding. The reasoning mechanism operates on a thesaurus hierarchy using the meaningful concepts therein for interactions between the target and source of metaphors. The methodology has been fully implemented and tested on human subjects. A Sun computer using ART and LISP was used.

The remaining sections of this paper elaborate upon the methodology. Section two discusses the terminology used. Section three presents the computational model as exemplified by the TisS program. Section four describes a test of the methodology, section five presents a discussion, and section six concludes the body of the paper.

2 Terminology

2.1 Metaphor and Analogical Reasoning

The concept of metaphor as used in this paper is the pattern **T**arget **is S**ource (**TisS**) such as in *The eye*

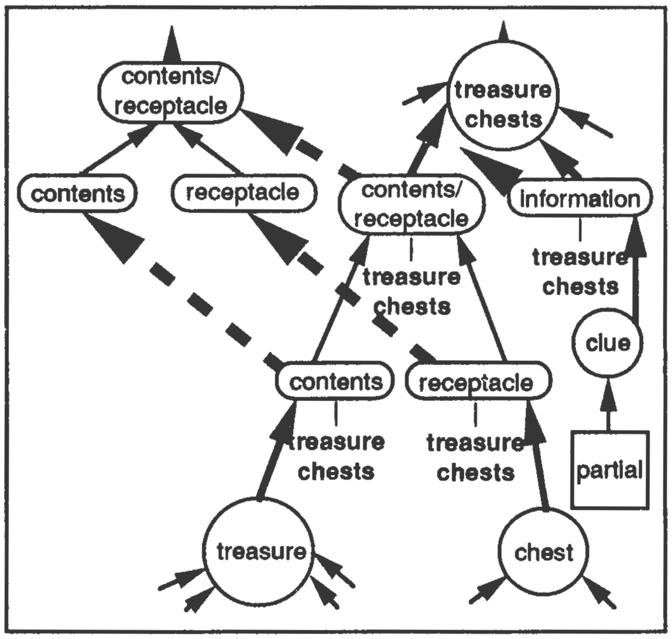


Figure 1: Partial Representation of *treasure chests* and Thesaurus (left)

is a camera or *Books are treasure chests* or *Criticism is a branding iron.* Analogical reasoning is the process involved in making sense of metaphors. In the case of a computer program, analogical reasoning results in changes in the structure (rearrangements and additions) to the target concept and rearrangements to the source concept (see discussions of interaction theory[Black, 1962; Tourangeau and Sternberg, 1982; Hunt, 1989]).

2.2 Knowledge Structures

There are two major structures in the design. One is the changeable conceptual network which captures knowledge about target and source domains. The other is a fixed thesaurus taxonomy which embodies abstract schemata.

2.2.1 Schemata, Gestalts, and Concepts

Three types of units are used in conceptual networks: schemata, gestalts, and concepts. *Schemata* are common units found within a concept which form the bridge between concepts within the target/source network through the fixed thesaurus. An example in Figure 1 is *content.* Schemata contain further components, which are either gestalts or concepts.

The *gestalts* only have a name and are found only as a final units (leaves) in a conceptual network. The only gestalt in Figure 1 is *partial.*

Concepts are composed of schemata which in turn consist of gestalts and other concepts. This results in the concept being a network of all three units. One concept in Figure 1 is *treasure chests.* Each of two different concepts has its own schemata but can be connected to the other concept through common schemata in the thesaurus.

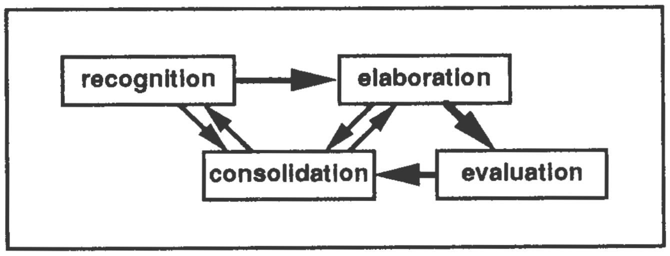


Figure 2: The Four Phases of the TisS Program

2.2.2 Thesaurus

The choice of schemata began with the work of Mark Johnson and George Lakoff. They discussed a handful of schemata at length[Johnson, 1987; Lakoff, 1987], enumerated 30, but considered the list to be incomplete.

To establish a stronger foundation for delimiting schemata with sufficient generality and scope, the two classes of **ABSTRACT RELATIONS** and **SPACE** from *Roget's Thesaurus of English Words and Phrases* [1987] were used. Besides its long-standing and continuing use, strong empirical support for exploring this semantic organization comes from the analysis of an earlier *Roget's Thesaurus* by Sedelow and Sedelow [1986]. A further analysis[Hunt, 1989] reveals that all but one (*compulsion*) of Johnson and Lakoff's schemata can be found at the category level of these two classes.

A further reason for using the thesaurus is its ability to serve as a model for the conceptual base theory. A conceptual base provides a common ground for both the source and target. Support for this theory comes from several quarters. From philosophy, this notion is called abstraction[Hesse, 1966; Darden and Rada, 1988; Way, 1988]. In analogical reasoning programs[Greiner, 1988; Burstein, 1988; Leishman, 1989], the use of abstractions is common. From psychology, studies in proverb comprehension[Honeck et al., 1980; Hoffman and Honeck, 1987], categorization[Honeck et al., 1987], and metaphorical representation[Honeck and Kibler, 1985] support the conceptual base theory.

A taxonomy with 182 abstract relation schemata and 134 space schemata are formed from the two classes. This remains stable while connected to the changing conceptual networks via links to instances of the schemata found within the concepts.

3 The Computational Model of Metaphor

3.1 General Reasoning Process

Several researchers in machine learning have studied analogical reasoning and use similar phases in the process[Kedar-Cabelli, 1985; Hall, 1986; Hall, 1989; Falkenhainer, 1987]. A study of the literature on creativity and discovery[Wallas, 1926; Hadamard, 1945; Parnes et al., 1977; Langley and Jones, 1986] also shows that phases parallel to these are in the creative problem solving process. From this research have come the four phases of the TisS Program (see Figure 2). The primary sequence involves:

source-cluster	a limit to the size of the source cluster
target-glimpse	first thoughts about the target
context	the current context of thinking
general-schema	general concepts useful in understanding metaphors

Figure 3: Recognition Phase Rules

interaction-size	size limitations for the interaction cluster
matching	criteria for matching clusters
transference	minimum count for node transference
modification	target modifications

Figure 4: Elaboration Phase Rules

recognizing relevant schemata (thesaurus entries) within the source structure

elaborating the target structure based on

- interaction with the source through common schemata
- otherwise significant schemata (testable)

evaluating the schemata using human judgement

consolidating the changes

Other pathways for program flow also exist among the phases. One provides more interaction between the source and target (Elaboration to Recognition), one consolidates source changes (Recognition to Consolidation to Recognition) and the third consolidates target changes (Elaboration to Consolidation to Elaboration).

3.2 Recognition

Given the metaphorical structure **Target is Source**, what is meant by recognition is locating a cluster of nodes for use in the subsequent steps. A node is one of schema, gestalt, or concept. The cluster must include the **Source**. The cluster is determined by parameterized rules (see Hunt [1989] regarding rule choice). These constrain a search that starts at the **Source** and spreads outward including acceptable nodes. Four rules are used to extract information from the source concept during the recognition phase. These are listed in Figure 3. An example of a general-schema rule is: (rec-rule SE-Contents 1 6) which states that the *contents* schema has priority 1, with a search range of 6 levels beyond the source.

3.3 Elaboration

Once the **Source** cluster is formed, it must interact with the **Target** to form an **interaction** cluster that eventually becomes part of the final **Target** cluster. Again, rules constrain the construction of the clusters. These are listed in Figure 4. An example of a modification rule is: (modify-target-rule QCj-Part 1 3) which states that the *part* schema can be added to the target cluster along with 1 ajoining level of nodes provided it is placed no more than 3 levels from the target concept.

3.4 Evaluation

The evaluation phase of TisS requires humans. For problem-solving metaphors getting the calculable answer is acceptable[Greiner, 1988]. To handle a wide range of metaphors, meaning will have to involve humans more closely. Section four discusses the evaluation phase including human testing.

3.5 Consolidation

There are two aspects to consolidation within the TisS design: rearrangement and modification. Rearrangement just determines which nodes, target or source, will be examined first on future passes. Modification involves adding new nodes to the target.

After all possible firings of the general schema rules are made for a given level of the source cluster, then the source cluster is rearranged leaving all matched nodes in front of all unmatched ones.

For the target cluster, modifications are the result of the firing of modification rules. Rearrangement occurs after modifications are performed, with new nodes and recognized nodes (i.e., fired matching rules) occurring first.

3.6 Process Overview

In Figure 5 the TisS process is depicted. The solid arrows indicate the flow of the program, heavier arrows being the major flow. The dashed arrows indicate changes to a cluster, heavy dashed arrows are creations or modifications while the light dashed arrows are rearrangements. After a metaphor is processed by TisS, both source and target clusters are typically rearranged and the target cluster may have new nodes. The evaluation phase also suggests changes in the rule priorities. All these changes provide a difference in the knowledge base for future metaphors.

4 The Test

Four metaphors (see Figure 6) were randomly chosen from a list of 260 nonliterary metaphors[Katz et al., 1988]. Each of the targets and sources was encoded into a network of nodes attached to the thesaurus (see Figure 1) based on lists of facts and ideas for each concept. For each target/source pair, two human subjects were enlisted, one to create each list. Neither subject was aware of the other concept (nor of the shared metaphor).

The TisS program was given each metaphor (and its reverse) and generated interpretations of each based on existing or newly formed connections to the thesaurus. The number of statements generated for each metaphor ranged from three to nine for a total of 40. Figure 7 shows selected interpretations and associated schemas.

Four human subjects then rated these 40 statements on scales of aptness and agreement. The five-valued *aptness* scale ranged from poor to good. With the subjects writing down their own interpretations of the metaphors, the five-valued *agreement* scale included the extremes of "no agreement" and "wrote the same interpretation" with "thought of the computer's interpretation" in the middle.

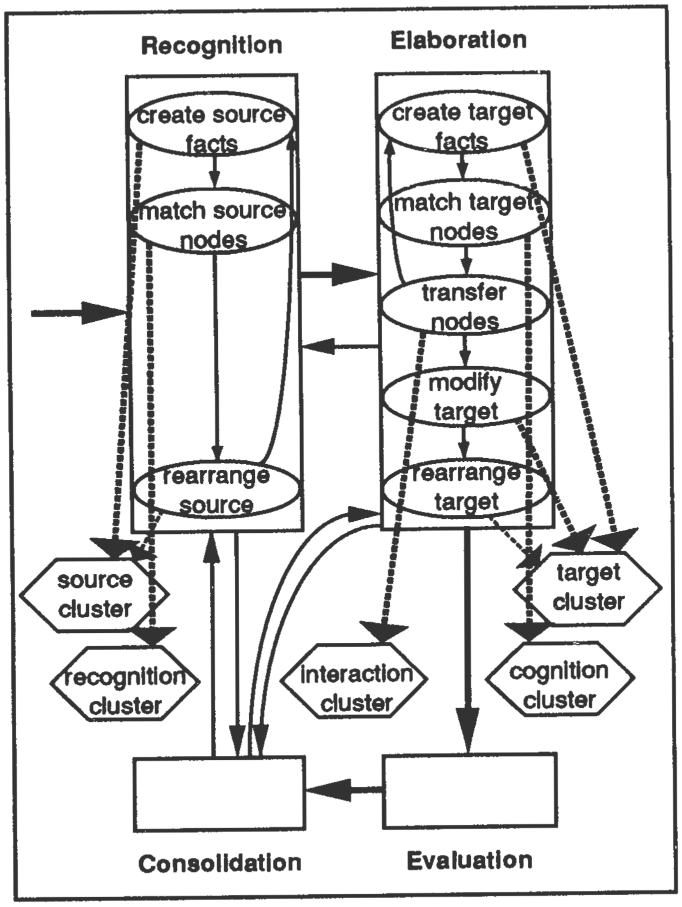


Figure 5: The TisS Process

Target		Source
the wind	is a	*cat*
books	are	*treasure chests*
mosquitoes	are	*vampires*
caves	are	*pockets*

Figure 6: Metaphors Used in Test

Statement	Schema
high (aptness and agreement)	
Books are receptacles with contents.	contents/receptacle
Mosquitoes are active in the evening.	evening
Blood is food for vampires.	food
low (aptness and agreement)	
The wind is small.	littleness
The cat is an agent.	agency

Figure 7: Selected Metaphors and Associated Schemata

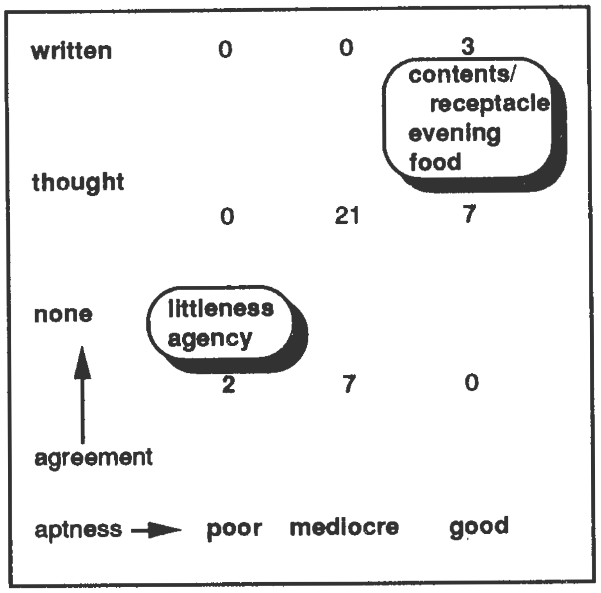


Figure 8: Subject Ratings (and Schemata) of TisS Statements about Metaphors

Each of the 40 statements was given a rating on a three-valued version of each scale based on unanimous ratings by the subjects for extreme values. Figure 8 shows the numerical results along with the associated schemata of the extreme values.

5 Discussion

Figure 8 shows that the three statements (of the 40) associated with *container/receptacle*, *evening*, and *food* were not only considered quite apt, but were in agreement with the written interpretations of the subjects. Another seven were as well considered good. At the other extreme two statements associated with *littleness* and *agency* were neither thought by any subjects nor considered at all apt. The most interesting statements would be those considered very apt but not even conceived of by subjects. These would be novel interpretations. There were none of these in this sample. Of the significant schemata, only *evening* was the result of TisS adding a new node.

The Recognition and Elaboration Phases are generally rule-governed breadth-first searches and model a domain interaction theory[Black, 1962; Tourangeau and Sternberg, 1982; Kelly and Keil, 1989]. The most demostrative action of TisS is the addition of new nodes to the target domain. These are subsequently evaluated by subjects to determine which schemata are worth adding. For early tests of metaphors, a broad if not complete range of schemata should be candidates for addition.

The Evaluation Phase requires human judgement. The broader range of metaphors intended for study require more human input, at least for now. The actual evaluation involved three steps on different days, including a test for consistency and the use of a novelty scale. It would be difficult to sustain the involvement of larger groups of subjects in a similar design.

During the Consolidation Phase, the target schemata are rearranged, with those matching source schemata moving to earlier positions. Added schema may also appear within the target. True to the interaction theory but not described in other analogical reasoning programs, the source also changes, via the rearrangement of schemata.

6 Conclusions

This paper presents a methodology for determining the semantics needed for a general analogical reasoning program. This methodology uses a broad base of concepts from a thesaurus. Initial testing was carried out and gave support to the methodology.

The next major step is to test new domains. There are already two new domains coded and fourteen others for which facts have been collected. It then remains to process the other metaphors from the source collection.

Revising the thesaurus is currently being studied. Two promising areas of revisions are part-whole relationships[Winston et al., 1987] and modes of existence[Hirst, 1989].

A continuing concern is improving the *goodness and independence* of each of the source and target representations.

Acknowledgements

Thanks to Tom Carey for his support.

References

[Black, 1962] Max Black (1962). Metaphor. *Models and Metaphors* by Max Black, editor. Cornell University Press, Ithaca, NY.

[Burstein, 1988] Mark H. Burstein (1988). Incremental Learning from Multiple Analogies. *Analogica* by Armand Prieditis, editor. pp37–62. Morgan Kaufmann Publishers, Inc., Los Altos, CA.

[Darden and Rada, 1988] Lindley Darden and Roy Rada (1988). Hypothesis Formation via Interrelations. *Analogica* by Armand Prieditis, editor. pp109–127. Morgan Kaufmann Publishers, Inc., Los Altos, CA.

[Eskridge, 1989] Thomas Eskridge (1989). Principles of Continuous Analogical Reasoning. *Journal of Expermental and Theoretical Artificial Intelligence.* v1 pp179–194.

[Falkenhainer, 1987] Brian Falkenhainer (1987). An Examination of the Third Stage in the Analogy Process: Verification-based Analogical Learning. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence.* pp260–263. Morgan Kaufmann Publishers, Inc., Los Altos, CA.

[Greiner, 1988] Russell Greiner. (1988, May). Learning by Understanding Analogies. *Artificial Intelligence.* v35 n1 pp81–125.

[Hadamard, 1945] Jacques Hadamard (1945/1954). *The psychology of Invention in the Mathematical Field.* Dover Publications, Inc., NY.

[Hall, 1986] Rogers P. Hall (1986). Understanding Analogical Reasoning: Computational Approaches. Technical Report 86-11. Irvine Computational Intelligence Project, Department of Information and Computer Science, University of California, Irvine, CA.

[Hall, 1989] Rogers P.Hall (1989, May). Computational Approaches to Analogical Reasoning: A Comparative Analysis. *Artificial Intelligence.* v39 n1 pp39–120.

[Helman, 1988] David H. Helman, editor (1988). *Analogical Reasoning, Perspectives of Artificial Intelligence, Cognitive Science, and Philosophy.* Kluwer Academic Publishers, Norwell, MA.

[Hesse, 1966] Mary B. Hesse (1966). *Models and Analogies in Science.* University of Notre Dame Press, Notre Dame, IN.

[Hirst, 1989] Graeme Hirst (1989). Ontological Assumptions in Knowledge Representation. *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning.* pp157–169. Morgan Kaufmann Publishers, Inc., Los Altos, CA.

[Hoffman and Honeck, 1987] Robert R. Hoffman and Richard P. Honeck (1987). Proverbs, Pragmatics, and the Ecology of Abstract Categories. *Cognition and Symbolic Structures: the Psychology of Metaphoric Transformation*, by Robert E. Haskell, editor. pp121–140. Ablex Publishing Corporation, Norwood, NJ.

[Honeck and Kibler, 1985] Richard P. Honeck and Clare T. Kibler (1985). Representation in Cognitive Psychological Theories of Figurative Language. *The Ubiquity of Metaphor: Metaphor in Language and Thought* by Wolf Paprotte and Rene Dirven, editors. pp381–423. John Benjamins Publishing Company, Amsterdam.

[Honeck et al., 1987] Richard P. Honeck, Clair Kibler, and Michael J. Firment (1987). Figurative Language and Psychological Views of Categorization: Two Ships in the Night? *Cognition and Symbolic Structures: the Psychology of Metaphoric Transformation* by Robert E. Haskell, editor. pp103–120. Ablex Publishing Corporation, Norwood, NJ.

[Honeck et al., 1980] Richard P. Honeck, Katherine Voegtle, Mark A. Dorfmueller, and Robert R. Hoffman. (1980). Proverbs, Meaning, and Group Structure. *Cognition and Figurative Language* by Richard P. Honeck and Robert R. Hoffman, editors. pp127–161. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ.

[Hunt, 1989] William T. Hunt (1989). *A Thesaurus-based Test-bed for Metaphorical Reasoning.* MSc thesis. University of Guelph, Guelph, Ontario, Canada.

[Johnson, 1987] Mark Johnson (1987). *The Body in the Mind.* University of Chicago Press, Chicago.

[Katz et al., 1988] Albert N. Katz, Allan Paivio, Marc Marschark, and James M. Clark (1988). Norms for 204 Literary and 260 Nonliterary Metaphors on 10 Psychological Dimensions. *Metaphor and Symbolic Activity.* v3 n4 pp191–214. Lawrence Erlbaum Associates, Hillsdale, NJ.

[Kedar-Cabelli, 1985] Kedar-Smadar T. Cabelli (1985, December). Analogy - from a Unified Perspective. Technical Report ML-TR-3. Department of Computer Science, Laboratory for Computer Science Research, Rutgers University, New Brunswick, NJ.

[Kelly and Keil, 1989] Michael H Kelly and Frank C. Keil (1987). Metaphor Comprehension and Knowledge of Semantic Domains. *Metaphor and Symbolic Activity.* v2 n1 pp33–51. Lawrence Erlbaum Associates, Hillsdale, NJ.

[Lakoff, 1987] George Lakoff (1987). *Women, Fire, and Dangerous Things.* University of Chicago Press, Chicago.

[Langley and Jones, 1986] Pat Langley and Randolph Jones (1986). A Computational Model of Scientific Insight. Technical Report 87-01. Irvine Computational Intelligence Project, Department of Information and Computer Science, University of California, Irvine.

[Leishman, 1989] Debbie Leishman (1989). Analogy as Constrained Partial Correspondence Over Conceptual Graphs. *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning.* pp223–234. Morgan Kaufmann Publishers, Inc., Los Altos, CA.

[Lenat et al., 1986] D. Lenat, M. Prakash, and M. Shepherd (1986, winter). CYC: using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks. *AI Magazine.* v6 n4 pp65–85.

[von Noppen et al., 1985] J.P. van Noppen, S. DeKnop, and R. Jongen, compilers (1985). *Metaphor: a Bibliography of Post-1970 Publications.* John Benjamins Publishing Company, Amsterdam.

[Parnes et al., 1977] Sidney Parnes, Ruth Noller, and Angelo Biondi (1977). *Guide to Creative Action.* pp146–151. Charles Scribner's Sons, NY.

[Prieditis, 1988] Armand Prieditis, editor (1988). *Analogica.* Morgan Kaufmann Publishers, Inc., Los Altos, CA.

[Roget, 1987] *Roget's Thesaurus of English Words and Phrases* (1987). Betty Kirkpatrick, editor. Longman Group UK Limited, Essex, England.

[Sedelow and Sedelow, 1987] Sally Sedelow and Walter Sedelow (1987). Semantic Space. *Computers and Translation* v2 pp235–245.

[Shibles, 1971] W. Shibles (1971). *Metaphor: An Annotated Bibliography and History.* Language Press, Whitewater, WI.

[Tourangeau and Sternberg, 1982] R. Tourangeau and R. J. Sternberg (1982). Understanding and Appreciating Metaphors. *Cognition.* v11 pp203–244.

[Wallas, 1926] Graham Wallas (1945/1926). *The Art of Thought.* C. A. Watts and Co. Ltd, London.

[Way, 1988] Eileen Cornell Way (1988). Dynamic Type Hierarchies: an Approach to Knowledge Representation through Metaphor. Phd thesis. Department of Philosophy, State University of New York at Binghamton.

[Winston et al., 1987] Morton E. Winston, Roger Chaffin, and Douglas Herrmann (1987). A Taxonomy of Part-Whole Relations. *Cognitive Science.* v11 pp417–444.

[Wolstencroft, 1989] John Wolstencroft (1989). Restructuring, Reminding, and Repair: What's Missing from Models of Analogy. *AI Communications.* v2 n2 pp58–71.

A SIMPLE ALGORITHM FOR LEARNING A FINITE-STATE MACHINE

Sukhamay Kundu
Computer Science Department
Louisiana State University
Baton Rouge, LA 70803, USA

ABSTRACT
We present a simpler and more efficient algorithm for learning a finite-state machine than the previously known methods. The present algorithm constructs the minimal deterministic finite-state machine and is based on a series of 'yes'/'no' answers to the queries of the form equivalent(x_1, x_2) and accepted(x), where x, x_1, and x_2 are input strings over the alphabet of the machine. If the minimal machine has m states and the size of the input alphabet is n, then the algorithm uses at most $O(m^2.n)$ queries. If we restrict the queries only to the form accepted(x), then the learning algorithm requires $O(m.n^{m-1})$ queries.

1. INTRODUCTION

One of the main criteria for a machine M_0 to be considered intelligent is that it should be able to discover algorithms for solving a given class of problems from a suitable specification of that problem class. Since a general algorithm corresponds to a Turing machine, it means that an intelligent machine should be able to construct a Turing machine M_P from the specification of a problem class P. In a sense, this is always possible if P is specified in a special form. For example, suppose the problem consists of deciding whether a given input string belongs to a language (class of strings) L_P or not. If L_P is specified by a regular expression, then the desired machine M_P is a finite-state machine that accepts the language L_P and there is a Turing machine M_0 (i.e., an algorithm) which can generate the transitions of such an M_P. More generally, suppose the problem class P is modeled by a primitive recursive function $f_P: N^k \to N = \{0, 1, 2, ...\}$, and the function f_P is specified by a finite set of expressions which define f_P in terms of certain initial functions, and the operations of function composition, primitive recursion, and unbounded minimalization [1].[§] It is not hard to design a Turing machine M_0 which can generate the transitions of a Turing machine M_P for computing the function f_P from such a specification. The learning problem is thus reduced to the problem of finding a suitable representation and, moreover, the learning problem becomes more or less trivial, given such a representation. However, we do not have at present an algorithm for constructing a representation in the form of regular expressions or function expressions using the initial functions, etc. from other forms of specification. This calls for a different approach to the learning problem. One such approach is called *learning from examples*, where one examines a set of input-output pairs of the machine M_P and tries to construct M_P from that information. In the remainder of the paper, we consider the problem of learning a finite-state machine from examples.

The class of finite-state machines are particularly important because it models the simplest form of recursively defined languages (concepts). The concept defined by a boolean function is non-recursive and is suitable only when the concept distinguishes among a finitely many alternatives ($\leq 2^n$ for a function of n boolean variables).

In practice, we do not often have a sufficient knowledge of a problem class P (equivalently, the associated set of strings L_P) to express the set L_P in the form of a regular expression. Instead, we generally have only a set of examples E of input-output pairs of M_P for the

§ The three commonly used initial functions are: (1) the successor function $\sigma(n) = n + 1$ for $n \geq 0$, (2) the 0-place function $\zeta(\ ) = 0$, which has no argument, and (3) the projection functions $\pi^k{}_i(n_1, n_2, ..., n_k) = n_i$ for all $n_1, n_2, ..., n_k \geq 0$.

language L_P. Here, an input-output pair has the form (x_i, 'yes') or (x_i, 'no'), depending on whether the input string $x_i \in L_P$ or not. A pair of the form (x_i, 'yes') is called a *positive* example and a pair of the form (x_i, 'no') is called a *negative* example. The learning task is to construct a finite-state machine which is consistent with a given set of examples E. Such a machine may be taken as an approximation to the actual machine M_P. Unfortunately, the class of finite-state machines is too large to identify its members in terms of their behavior on a given finite set of example inputs. There are in fact an infinite number of non-equivalent (i.e., which differ in the output for at least one input other than those in E) finite-state machines which are consistent with a given finite set of examples E. Thus one cannot determine M_P exactly from any finite set of examples E no matter how simple or complex the machine M_P is. The situation does not improve much even if we have additional information about the regular language L_P such as that L_P is disjoint from (or contains) another known regular language L′, where L′ may be given in the form of a regular expression, say. Thus in order to learn M_P exactly the learning algorithm must have access to a teacher (oracle) which can answer the queries of type (1)-(4) shown below or some other equivalent forms of them. We say that the input strings x_i and x_j are *equivalent*, denoted by $x_i \approx x_j$, if for each string y, either both x_i.y and x_j.y are accepted by M_P or both are rejected by M_P [1, 2].

(1) Is the machine M_P to be learned is equivalent to the one currently constructed by the learning algorithm?

(2) Give new examples of input-output pairs (x_i, y_i), or equivalently, determine the output y_i = 'yes'/'no' = accepted(x_i) for an input x_i, where x_i may be generated by the learning algorithm.

(3) Are the input strings x_i and x_j equivalent?

(4) Is the number of states in $M_P \leq$ m?

The learning algorithm in [3] is based on the queries of type (1)-(2). The main part of that algorithm constructs a deterministic finite-state machine M'_P which is consistent with a given set of examples E. If M'_P is not equivalent to the machine M_P, then it obtains one or more other inputs x_i from the teacher such that x_i is accepted by one of M'_P and M_P and rejected by the other. The algorithm then modifies M'_P on the basis of the new example x_i, and repeats the cycle until the teacher confirms that M'_P is equivalent to M_P. The finite-state machine M'_P constructed at each stage has the smallest number of states and each new M'_P obtained has the same output for a larger set of input strings than that for the previous M'_P.

We give two algorithms LEARN and LEARN2 for learning a finite-state machine. The first algorithm LEARN uses the queries of type (2) and type (3). The second algorithm LEARN2 is a slight variation of LEARN, where only the queries of type (2) are used, in addition to a bound on the number of states in M_P (queries of type (4)). Note that the queries of type (3) are, in a sense, implicit in the queries of type (1). In this sense, the algorithm LEARN does not assume any additional help from the teacher than the algorithm in [3]. In LEARN2, the bound on the number of states helps to replace the queries on equivalence of strings. We should point out that although the queries of type (3) simplifies the learning of the finite-state machine M_P, the construction of the machines M'_P given in [3] is of much independent interest. This is because, from the practical point of view, it is important to be able to construct a finite-state machine which is consistent with a given set of examples E and which has the minimum number of states.

2. THE NEW LEARNING ALGORITHM

The basis of the new learning algorithm is the Myhill-Nerode's theorem given below [1]. The equivalence relation "$\approx$" on the set of input strings of a finite-state machine M_P has the following properties (i)-(iii). Here, Σ is the alphabet of M_P.

(i) For any two strings x_1 and x_2, $x_1 \approx x_2$ implies that $x_1.\alpha \approx x_2.\alpha$ for all $\alpha \in \Sigma$. (Repeated application of this gives x_1.y $\approx$ x_2.y for all input strings y.)

(ii) The equivalence relation "$\approx$" has only a finitely many equivalence classes.

(iii) The set of strings accepted by M_P is the union of zero of more of these equivalence classes.

Given the equivalence classes, one can define a finite-state machine M^*_P, which is equivalent to M_P (i.e., accepts the same strings) as follows. Let $[x_1]$, $[x_2]$, ..., $[x_k]$ be the equivalence classes of the set of input strings Σ^*. We define one state s_i in M^*_P for each equivalence class $[x_i]$. The starting-state corresponds to the class $[x_1]$, say, which contains the empty string λ. A state s_i is a final state of M^*_P if and only if the equivalence class $[x_i]$ contains a string which is accepted by M_P (and hence each string in $[x_i]$ is accepted by M_P by the property (iii)). Finally, we have the transition $\delta(s_i, \alpha) = s_j$ in M^*_P if and only if the string $x_i.\alpha \in [x_j]$. The property (i) assures that the transitions are well-defined. The machine M^*_P has the important property that it has the smallest number of states among all finite-state machines which are equivalent to M_P. In particular, M^*_P is unique except for the names of the states.

Since there is no way to distinguish the machine M_P from its equivalent minimum machine M^*_P in terms of the input-output behavior, it is not surprising that a learning algorithm for a finite-state machine determines (both in our algorithm and that in [3]) only the equivalent form M^*_P, which is after all the simplest (equivalent) form of M_P.

We now present the first learning algorithm LEARN. We denote the current set of states by STATES. A state $s_x \in$ STATES, corresponding to the equivalent class [x], is said to be *closed* if all the transitions from that state have been determined; otherwise, s_x is said to be *open*. The current set of open states is denoted by OPEN. In Step(2) of LEARN, the states $s_x \in$ OPEN may be selected in an arbitrary order. The algorithm stops when the set OPEN becomes empty.

Example 1. Table 1 illustrates the computations of LEARN in learning the finite-state machine M_P shown in Fig. 1. The input alphabet of M_P is $\Sigma = \{0, 1\}$ and M_P accepts the strings which have an even number of 0's and an even number of 1's [3]. In Table 1, we show the successive choice of x, the set OPEN at the end of each iteration of Step (4), the new state added to the set STATES (if any), and the transition defined. We write the state s_x simply as [x]. If we select the successive states s_x in Step (2) in the order in which they are added to OPEN (i.e., in the non-decreasing order of lengths), then the underlying search-tree for the new states of M^*_P takes the form as shown in Fig. 2(i). This corresponds to the breadth-first search for the states of M^*_P. Fig. 2(ii) shows the corresponding search-tree if the states s_x are selected in the depth-first fashion. Each branch of the tree from a parent node x to a child node y = x.α corresponds to the transition $\delta(s_x, \alpha) = s_y$, where α is the label of that branch. ■

It is easy to see that the algorithm LEARN takes at most $O(m^2.n)$ queries, where m = the number of states in M_P and n = the size of the alphabet Σ. For each state s_x, there is one query accepted(x) and at most (m.n) queries of the form equivalent(x.α, y), where s_y is one of the current

Algorithm LEARN (for learning a finite-state-machine):

Input: The input alphabet Σ of the finite-state machine M_P and a set of queries of the form accepted(x_1) and equivalent(x_1, x_2), where the strings x_1 and x_2 are generated by the algorithm.

Output: The minimal deterministic finite-state machine M^*_P which is equivalent to M_P.

```
1.  [Initialize.] Let STATES = {$s_\lambda$} = OPEN.
2.  If OPEN = empty-set, then the desired finite-state machine has been obtained, and stop. Other-
    wise, choose a state $s_x \in$ OPEN and delete it from OPEN.
3.  If accepted(x) = 'yes', then label $s_x$ as a final state.
4.  [Close $s_x$.] For each symbol $\alpha \in \Sigma$ do the following:
    (i)   Find the unique state $s_y \in$ STATES, if any, such that $s_{x.\alpha}$ is equivalent to $s_y$.
    (ii)  If no such state $s_y$ exists, then let y = x.α and add the state $s_y$ to both OPEN and
          STATES.
    (iii) Define the transition $\delta(s_x, \alpha) = s_y$.
5.  Go to Step (2). ■
```

TABLE 1. Computations performed by the algorithm LEARN in learning the finite-state machine in Fig. 1.

x	Final state	α	New state [x] or equivalent [y]	OPEN states	Transitions from the state [x]
			$[\lambda]$	$[\lambda]$	
$x = \lambda$	yes	$\alpha = 0$	$[0]$	$[0]$	$\delta([\lambda], 0) = [0]$
		$\alpha = 1$	$[1]$	$[0], [1]$	$\delta([\lambda], 1) = [1]$
x = '0'	no	$\alpha = 0$	$[00] \approx [\lambda]$	$[1]$	$\delta([0], 0) = [\lambda]$
		$\alpha = 1$	$[01]$	$[1], [01]$	$\delta([0], 1) = [01]$
x = '1'	no	$\alpha = 0$	$[10] \approx [01]$	$[01]$	$\delta([1], 0) = [01]$
		$\alpha = 1$	$[11] \approx [\lambda]$	$[01]$	$\delta([1], 1) = [\lambda]$
x = '01'	no	$\alpha = 0$	$[010] \approx [1]$	empty	$\delta([01], 0) = [1]$
		$\alpha = 1$	$[011] \approx [0]$	empty	$\delta([01], 1) = [0]$

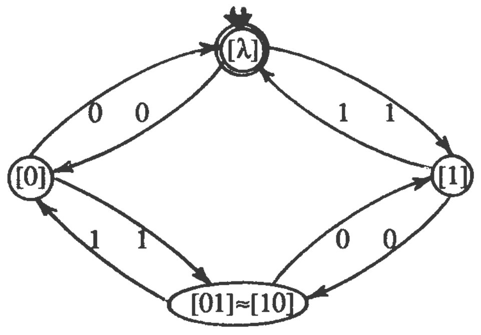


Figure 1. The state-diagram of the finite-state machine which accepts all strings over $\Sigma = \{0, 1\}$ containing an even number of 0's and an even number of 1's.

states. The time required by the algorithm LEARN is also at most $O(m^2.n)$.

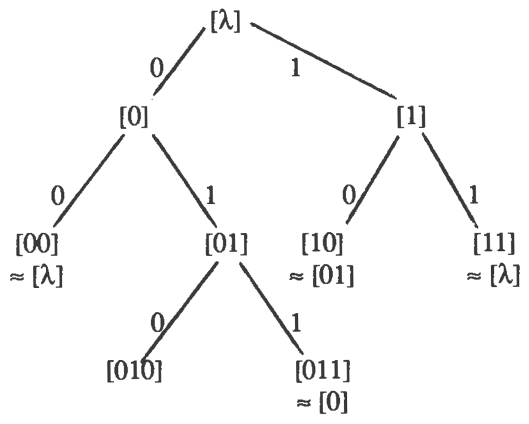
3. A MODIFIED FORM OF LEARN USING ONLY QUERIES OF TYPE (2)

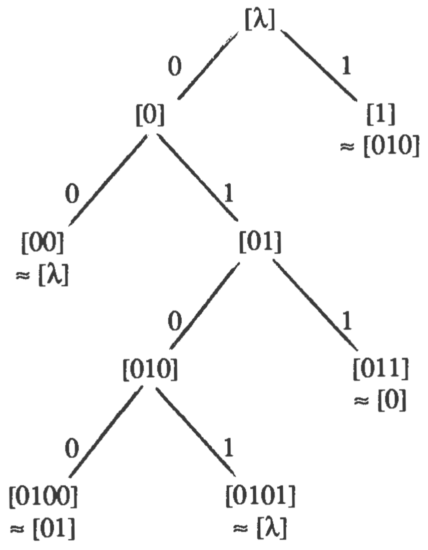
Actually, it is possible to learn a finite-state machine M_P using only queries of type (2) provided we are given an upper bound m on the number of states in M_P. In that case, one can determine the equivalence of two strings x_i and x_j by comparing the values of accepted(x_i.y) and accepted(x_j.y) for sufficiently many strings y. To be precise, we have the following lemma [1].

Lemma 1. If the finite-state machine M_P has m states, then two input strings x_i and x_j are equivalent if and only if for all strings y of length $\leq$ (m − 2), we have accepted(x_i.y) = accepted(x_j.y).

In [3], each approximation M'_P to the desired finite-state machine M_P is constructed on the basis of verifying the equality accepted(x_i, y) = accepted(x_j.y) for a certain set of strings y; each new approximation uses a larger class of strings y than that used in the previous approximations. We give below a variation of the algorithm LEARN, called LEARN2, where only the queries of type (2) are used, in addition to the upper bound m. The only difference between LEARN and LEARN2 is in Step 4(i); the modified form of this step is shown as Step 4(i′) below. We denote by TEST the set of all strings of length $\leq$ (m − 2). There are $(n^{(m-1)} - 1)/(n - 1)$ strings in TEST.

Example 2. Table 2 illustrates the computations of LEARN2 in learning the finite-state





(i) Breadth-first search-tree. (ii) Depth-first search-tree.

Figure 2. The underlying search-trees in learning the finite-state machine in Fig. 1 using breadth-first and depth-first search for new states. A node is a terminal node in the search-tree if it equivalent to a node in the current tree.

Algorithm LEARN2 (for learning a finite-state-machine using a bound on the number of states):

Input: The input alphabet Σ of the finite-state machine M_P, an upper bound m on the number of states, and a set of queries of the form accepted(x), where x is a string of length less than 2m.

Output: The minimal deterministic finite-state machine M^*_P which is equivalent to M_P.

```
1.  [Initialize.] Let OPEN = $\{s_\lambda\}$ and STATES = $\{s_\lambda\}$.
2.  If OPEN = empty-set, then the desired finite-state machine has been obtained, and stop. Otherwise, choose a state $s_x \in$ OPEN and delete it from OPEN.
3.  If accepted(x) = 'yes', then label $s_x$ as a final state.
4.  [Close $s_x$.] For each symbol $\alpha \in \Sigma$ do the following:
    (i') Obtain 'yes'/'no' values of accepted(x.α.z) for $z \in$ TEST and find the unique state $s_y \in$ STATES, if any, such that accepted(x.α.z) = accepted(y.z) for all $z \in$ TEST.
    (ii) If no such state $s_y$ exists, then let y = x.α and add the state $s_y$ to both OPEN and STATES.
    (iii) Define the transition $\delta(s_x, \alpha) = s_y$.
5.  Go to Step (2). ■
```

machine considered in Example 1. The set TEST has 7 strings TEST = {λ, 0, 1, 00, 01, 10, 11}. The middle part of Table 2 shows the values of accepted(x.α.z) for $\alpha \in \Sigma$ and $z \in$ TEST. Two strings x_i and x_j are considered equivalent if the associated Y/N-row for x_i and x_j are the same. Thus λ is equivalent to 00 and 11, 01 is equivalent to 10, and so forth. The final finite-state machine constructed by LEARN2 is once again the minimal equivalent machine of M_P. ■

That the algorithm LEARN2 requires O(m.n^{m-1}) queries of the form accepted(x) is clear because TEST is of size O(n^{m-2}) and for each state [x] in $M^*{}_p$ one has to make O(n^{m-1}) many evaluations of queries of the form accepted(x.α.z). The time required by LEARN2 is at most O(m.n^{m-2}.(m + n)) because to test the equivalence of a state s_x one has to compare O(n^{m-2}) values of accepted(x.α.z) with each of the current states and $\alpha \in \Sigma$. The fact that the algorithm LEARN2 takes exponential amount of time in n is not surprising because the problem of determining a finite-state machine from input-output behaviors alone is known to be NP-complete [4].

Acknowledgement: I like to thank Dr. Jianhua Chen for many valuable technical discussions in course of the preparation of this paper.

4. REFERENCES

1. Lewis, H. R. and Papadimitriou, C. H., *Elements of the theory of computation,* Prentice-Hall, Englewood Cliffs, NJ, 1981.
2. Hopcroft, J. E. and Ullman, J. D., *Introduction to automata theory, languages, and computation,* Addison-Wesley Publ. Comp., Reading, MA, 1979.
3. Angluin, D., *Learning regular sets from queries and counterexamples,* Information and Computation, 75 (1987), pp. 87-106.
4. Gold, E. M., *Complexity of automaton identification from given data,* Information and Control, 37(1978), pp. 302-320.

TABLE 2. Computations performed by LEARN2 in learning the finite-state machine in Fig. 1.

x.α	TEST = {λ, 0, 1, 00, 01, 10, 11}							New state or	OPEN	Transitions from	
	λ	0	1	00	01	10	11	equivalent [y]	states	state [x]	
λ	Y	N	N	Y	N	N	Y	[λ]	[λ]		
0	N	Y	N	N	N	N	N	[0]	[0]	δ([λ], 0)	= [0]
1	N	N	Y	N	N	N	N	[1]	[0], [1]	δ([λ], 1)	= [1]
00	Y	N	N	Y	N	N	Y	[00] ≈ [λ]	[1]	δ([0], 0)	= [λ]
01	N	N	N	N	Y	Y	N	[01]	[1], [01]	δ([0], 1)	= [01]
10	N	N	N	N	Y	Y	N	[10] ≈ [01]	[01]	δ([1], 0)	= [01]
11	Y	N	N	Y	N	N	Y	[11] ≈ [λ]	[01]	δ([1], 1)	= [λ]
010	N	N	Y	N	N	N	N	[010] ≈ [1]	empty	δ([01], 0)	= [1]
011	N	Y	N	N	N	N	N	[011] ≈ [0]	empty	δ([01], 1)	= [0]

DATAX: A Framework for Machine Discovery of Regularity in Data*

Howard J. Hamilton
School of Computing Science
Simon Fraser University
Burnaby, B.C., Canada V5A 1S6
hamilton@cs.sfu.ca

Abstract

A framework for automatically discovering regularities in numerical data is described. The function-finding problem, a restricted version of the numerical data-analysis problem, is characterized as an induction problem. The framework is presented as an exponential-time algorithm, DATAX, which organizes and chooses among numerical-analysis techniques for automatic scientific discovery. The approach combines a theory-driven approach based on a set of function forms ("curve-fitting") with a data-driven approach based on relations observed in subsets of the data. A polynomial-time algorithm, DATAX-2, is provided for a restricted form of the problem where the target function is binary-partition decomposable. DATAX-2 is more efficient than the Reduction method [Wu 88], but applicable to the same class of problems.

1 Introduction

A framework is described for searching for regularities in numerical data. This framework combines techniques developed in AI for controlling automatic discovery with traditional numerical-analysis ("curve-fitting") techniques. The framework is presented as an algorithm, DATAX, which requires exponential time in the worst case. A polynomial-time algorithm, DATAX-2, is then provided for a restricted form of the problem. The framework is appropriate to observational data that have been accumulated and need to be analyzed for numerical relationships. DATAX-2 is more efficient than the methods used in BACON [Langley 78; Langley *et al* 87], ABACUS [Falkenhainer and Michalski 86], and Reduction [Wu 88; Wu and Wang 89].

Both numerical-analysis and AI methods have their weaknesses. Often in data analysis, a number of numerical-analysis techniques must be applied, but the choice and ordering of the techniques is not automated and is often done in an ad hoc way. Two weaknesses in previous AI discovery programs have been the inability to handle noise in data and inefficiency in searching for regularities in numerical data. Incorporating numerical-analysis techniques into a control strategy derived from an AI discovery program seems to be a promising way of overcoming both of these weaknesses. In this paper, the second of these weaknesses is addressed by showing how the efficiency of an AI discovery program can be improved by incorporating numerical-analysis techniques. Numerical-analysis techniques improve the efficiency by searching for classes of relations between variables instead of individual relations. As a simple example, DATAX considers all functions of the form $f(x) = x^i$ at once rather than considering each of $f(x) = x, f(x) = x^2, f(x) = x^3, \ldots$ separately, as BACON does.

In the *numerical-data analysis problem*, an attempt is made to find sets of inequalities and/or equations that are consistent with numerical data. The *function-finding problem* is a restricted version of the numerical-data analysis problem in which each hypothesis is an equation between a numerical constant and a nontrivial function on all the variables (a nontrivial function on *n* variables cannot be redefined using a proper subset of the variables). Such an equation is called a *target equation*, and the included function is called a *target function*. Thus, the solution to a function-finding problem is a nontrivial function on a set of variables that yields a constant value for all data tuples. For example, for a sequence of (x,y,z) data tuples including $\{(1,0,1), (2,3,1), (3,2,2)\}$, a suitable target function is $x^2 - yz^2$, which has a constant value of 1 for these data tuples. The function-finding problem has been attempted by BACON.1, BACON.3, BACON.4, ABACUS, and Reduction. In [Langley *et al.* 87], the place of the function-finding problem in the process of scientific discovery is identified.

The DATAX method is applied to the function-finding problem. In this method, the task of discovering a target equation on *n* variables is reduced to discovering simpler equations on fewer variables. Given a set of *n* variables, the DATAX method first finds a *constant-valued function*, a nontrivial function on a subset of two or more of the variables that yields a constant value for all examined data tuples. Then the subset of the variables is replaced by a newly created *synthetic variable* defined to have the value given by applying the constant-valued function to the subset of variables. This process is repeated until only a single variable remains. Beginning with the final variable,

* This work was supported by a Simon Fraser University Graduate Fellowship, a B.C. Advanced Systems Foundation Graduate Research Scholarship, and Natural Sciences and Engineering Research Council of Canada Grant A4309.

the synthetic variables are replaced by their definitions in terms of constant-valued functions until the target function is produced. (Since the substitution is straightforward, it will not be considered further.)

The remainder of the paper is organized as follows. The problem of finding regularity in data is characterized as an induction problem in Section 2. In Section 3, Algorithm DATAX is presented and discussed. In Section 4, one particular class of functions for which the method is efficient is identified, and an efficient algorithm for this class, DATAX-2, is presented, along with its complexity analysis and suggestions for improving its efficiency in practice. Finally, in Section 5, the method is compared to other published methods and conclusions are drawn.

2 Function Finding as Induction

In this section, the function-finding problem is characterized as an induction problem. Before doing so, induction is defined and data-driven and theory-driven approaches to induction are described.

2.1 Induction

Induction is the process of forming a general concept on the basis of specific examples. Adapting slightly from [Simon and Lea 74], all possible examples of the concept form an *example space* and all hypotheses that are considered form a *hypothesis space*. For clarity, sometimes the examples are assumed to be presented by a *presenter* as requested by an *inducer*, who is conducting a structured search of the hypothesis space. To solve an induction problem, the inducer must identify the *target hypothesis*, a hypothesis in the hypothesis space that matches all examples in the example space. The approach taken in this paper involves (1) making assumptions about the nature of the example space so that only a small portion of it needs to be examined, and (2) searching a restricted hypothesis space that matches these assumptions.

According to [Langley *et al.* 87] (pp.13-14), it is assumed that when scientific discovery is attempted, a body of data and/or some scientific laws are presumed to be already known. If the set of previously known laws is empty, the discovery process is called *data driven*. On the other hand, if the set of data is empty, the discovery process is called *theory driven*. The terms *data-driven induction* and *theory-driven induction* can be defined to describe the corresponding approaches to induction. Purely data-driven and purely theory-driven approaches represent extreme cases, and there is a continuum of approaches in between.

We claim that a purely data-driven approach is not possible. It is only by searching for a particular relation in the data, perhaps by applying one of a set of operators, that any relation is found. Thus, existing formulations of data-driven approaches also assume that a set of base-level operators is known and that some of these operators are relevant. For example, in one data-driven approach, BACON.1 [Langley 78; Langley *et al.* 87], the multiplication, division, and modulus operators are assumed to be relevant to discovering a target function in numerical data. In BACON.1, the hypotheses are not stated in terms of possible forms for the target function, but in terms of operators that might be applied to the data. The implicit theory is that any function will consist of a combination of these operators. In general, with a data-driven approach, each hypothesis is assumed to be a combination of the operators, but there may not be a concise description of the hypothesis space.

To gauge the portions of data-driven and theory-driven induction in a particular method, consider whether the search space is defined in terms of the examples (data) or the hypotheses (theories). The search employed in DATAX is an attempt to combine theory-driven and data-driven methods. In the theory-driven portion, each of a set of function forms is considered in turn. If a match is found between a function form and the data, a function is derived from the function form. This function is not chosen as the target function, as it would be in a completely theory-driven approach; instead, the function is used as one of the component functions from which the target function is composed. Thus, the function forms correspond to the operators in BACON.1.

2.2 The Function-Finding Problem

In numerical-data analysis, the examples are data tuples giving numerical values for a set of variables, and the hypotheses are sets of mathematical equations and inequalities involving the variables. In the function-finding problem, the hypothesis space is greatly restricted by assuming that there exists a nontrivial function on the complete set of variables that yields a constant value. Since an induction problem can be defined as determining whether or not a method is sufficient to choose a hypothesis which describes a set of examples, the function-finding problem is discussed in the next two subsections with respect to the *hypotheses* and the *examples*. The assumptions needed for the DATAX method are highlighted. DATAX itself is discussed in Section 3.

2.3 Hypotheses

The hypothesis space for the function-finding problem is the set of target equations between target functions (nontrivial functions on the complete set of variables) and constant values. We assume that each target function can be decomposed into functions. To describe the hypothesis space further, we consider the set of function forms and their relation to the process of decomposing the target function.

In the DATAX method, it is assumed that each function used in a decomposition can be produced by replacing the parameters in some function form. by constants. In Algorithm DATAX, P represents the complete set of function forms and P_i represents the set of function forms on i variables. For example, P_2 gives the set of function forms with two variables. Wu suggests four function forms for P_2, with variables x and y and parameters r_j and c_j [Wu 88]:

1. $f(x, y) = x^{r_1} + c_1 y^{r_2}$
2. $f(x, y) = x^{r_1} y^{r_2}$
3. $f(x, y) = (x^{r_1} + c_1)^{r_2} (y^{r_3} + c_2)$
4. $f(x, y) = (x^{r_1} + c_1)^{r_2} + c_2 (y^{r_3} + c_3)^{r_4}$

Choosing a small set P in this manner restricts the set of functions used in the composition of the target function to those with only a small set of forms. This assumption is not unusual in data analysis; for example, in statistical inference, one often begins with the assumption that the target function is a linear combination of the variables and small powers of the variables. As well, each variable is assumed to appear in only one of the functions. These two assumptions can have a crucial effect on DATAX's ability to find the "correct answer", i.e., the actual relation in the data. Because of the first assumption, if DATAX does not find any function, then either no nontrivial function on all the variables exists or the actual function does not correspond to a composition of the given function forms. Because of the second assumption, complicated function forms involving several variables are needed to handle cases where a single variable interacts with several variables. For both cases, the set of function forms can be expanded as necessary.

If a variable must appear in two different The second assumption limits the complexity of the variable combinations in the target function

2.4 Examples

Four relevant considerations for examples are their *type* (positive or negative), *format*, *presentation*, and *number*.

2.4.1 Type

A *positive example* is a valid instance of the target hypothesis; a *negative example* is a data tuple that is inconsistent with the target hypothesis. Since numerical-analysis techniques do not use negative examples, any available data tuple is assumed to be a positive example.

2.4.2 Format

Each example is described by a data tuple giving a value selected from a continuous interval for each relevant variable. Thus, it is assumed that each data tuple is complete. Typically, the units of the values are not given, which implies that they are unknown, irrelevant or nonexistent. This assumption is standard in data analysis, but in COPER [Kokar 86 88], the search for relevant variables is motivated by a dimensional analysis using the units of the target function (say $kg^1m^{-1}s^{-2}$) and the units of the known variables. The DATAX method could be modified to use similar consistency information if the search for a function was constrained by considering only the function forms involving compatible combinations of the variables' units.

2.4.3 Presentation

In general, the order in which examples are presented to an inducer can be *controlled* or *uncontrolled*. With *controlled example presentation*, the inducer has some degree of control over the order in which examples are presented. Control is exerted by placing constraints on the values for a subset of the variables in the next data tuple to be presented. A constraint either restricts a variable to have one of a set of values or not to have any of a set of values.

In numerical analysis, the data is assumed to be a given, fixed set; this corresponds to uncontrolled presentation because the inducer has no control over the order in which data tuples are presented. In AI discovery programs, it is assumed that presentation is controlled to some degree. For example, in BACON.1 the inducer can choose the values for a fixed set of variables. When the inducer is given control over only a subset of the variables, these variables are called the *predictor* or *independent* variables and the other variables are called the *response* or *dependent* variables. DATAX assumes that controlled presentation is available and that constraints may be placed on any of the variables.

In many experiments, the number of occurrences of a data tuple is significant, but, for simplicity, in DATAX it is not. To use DATAX in such situations, the format of the data tuples should be converted, either by including a value to uniquely identify each data tuple, or by discarding duplicates and adding a value to all remaining tuples to represent the number of occurrences. Such a conversion will prevent information about tuple occurrences from being lost.

2.4.4 Number

Although the example space is infinite, only a finite number of examples are accessed by DATAX. The number of data tuples accessed depends on the number of variables being considered and the number of data tuples required to match a function form (i.e., to choose values for parameters in a function form to produce a function). The number of examples required to match a function form depends on the matching method. In DATAX, $\#D$, the maximum number of tuples needed for any matching method, is specified in advance. For example, choosing $\#D = 40$ would limit numerical-analysis procedures used for matching to at most 40 data tuples. An alternate approach would be to include with each function form a specification of the number of tuples required to match it. As the first step of the procedure for matching a function form, the number of tuples available could then be checked and more obtained if necessary.

2.5 Summary of Assumptions

The following assumptions are required for the DATAX method:

- The target equation can be written in the form $f(V_1,...,V_n) = c$ where c is a numerical constant and function f is a nontrivial function on the variables $V_1,...V_n$.
- There exists a decomposition of function f such that each function used in the decomposition is an instance (with parameters given constant values) of a function form in a set of function forms P.
- Data tuples are complete.
- Controlled presentation of data tuples is available.
- Duplicate data tuples are not significant.
- A maximum number of data tuples $\#D$ needed to match any function form is known for the function forms in P.
- The data tuples are in general position, i.e., they are not chosen to disguise the target function.

3 DATAX Method

3.1 DATAX Algorithm

An outline of the DATAX method is as follows:

```
Set $S$ to the original set of variables.
while $S$ contains more than one variable:
    for all subsets $U$ of two or more variables in $S$,
    from smallest to largest subsets:
        if a constant-valued function $f$ on $U$ can be
        found
        then Remove the variables in $U$ from $S$;
            Create a new synthetic variable
            corresponding to $f$ and add
            it to $S$;
            Restart the for loop with the new value
            for $S$.
    if no constant-valued function is found for any
    subset
    then the method fails.
```

DATAX is easily seen to have exponential time complexity because in the worst case, all subsets of S (except those containing fewer than 2 elements) are examined. DATAX-2, the polynomial-time restricted version of DATAX given in Section 4.2, is obtained by restricting the size of the subsets to 2 and avoiding redundant matches.

Algorithm DATAX(V, P, Get_Data, #D).

(* Given a set V of n variables $\{V_1,...,V_n\}$, a set P of parameterized function forms, and access via Function Get_Data to data tuples providing values for these variables, this algorithm either finds and returns a target equation or returns **null** if no equation can be found. The number of data tuples to be used for matching against forms is given by #D. *)

```
        $c \leftarrow 0$
        $S \leftarrow V$
        while $|S|>1$
            $OldS \leftarrow S$
i-loop:     for $i = 2,3,..., |S|$
                for all $U \subseteq S$ such that $|U| = i$
                    $D \leftarrow$ Get_Data($V$,$S - U$,#$D$)
                    for all $p \in P_i$
                        $\{e,f\} \leftarrow$ Match_Form($p,U,D$)
                        if $f \neq$ null
                        then $c \leftarrow c + 1$
                            Make_Variable ($V_{n+c}$,$f$,$U$)
                            $S \leftarrow S - U + \{V_{n+c}\}$
                            exitloop($i$-loop)
            if $S = OldS$
            then return(null)
        return(e)
```

Algorithm DATAX includes calls to several functions:

1. *Function Get_Data*(V,C,k) obtains k data tuples on the variables in V by constraining the variables in C to constant values and leaving the other variables unconstrainted. The constant values for the variables in C are obtained from the first data tuple. If a synthetic variable is present in C, the variables in terms of which it is defined are also constrained to constant values.

2. *Function Match_Form* applies an appropriate method to try to match a form p to all data tuples in D. In particular, Match_Form tries to find values for the parameters of form p such that the resulting function f has a constant value on all data tuples in D. If successful, an equation e between f and the constant is created, and both f and e are returned; otherwise, **null** is returned for both.

3. *Function Make_Variable* defines a new synthetic variable (i.e., adds a new element to V) whose values are related to the values of the variables in U by function f. An implementation of Make_Variable would likely create a procedure for automatically calculating values for V_{n+c} as data tuples are accessed. At most $n-1$ synthetic variables will ever be defined because S starts with n variables and each time one variable is defined and added to S, at least two variables are removed.

A few points need to be made about the methods used for finding a constant function consistent with the set of data tuples and for updating the set of variables.

3.2 Finding a Constant-Valued Function

To find a constant-valued function on a subset U of the variables consistent the data tuples, a match is attempted between a sequence of function forms and these data tuples. Two relevant issues are (1) choosing the technique for matching and (2) choosing the function forms; these issues are now discussed in turn.

A match is obtained if the parameters of a function form can be given constant values such that the resulting function yields a constant value for all the tuples. A procedure for finding values for parameters is associated with each function form. A procedure may simply calculate a closed-form expression for each parameter, or it may apply a numerical-analysis technique to obtain values for all expressions. Where available, the closed-form expression provides the result with the least effort. For example, for a simple linear equation $y=ax+b$ and two data tuples (x_1,y_1) and (x_2,y_2), the constants can be calculated as $a=(y_2-y_1)/(x_2-x_1)$ and $b=y_1-ax_1$. For the second function form in P_2 given in Section 2.3, a similar closed-form expression can be derived by taking natural logarithms. However, since closed-form expressions are not available for many function forms, this method is not generally applicable.

The second method of finding values for parameters is to use iterative numerical-analysis techniques, such as Newton's Method. For such an approach, the values from the data tuples are used to create a set of equations (called "error functions") of the form $e(r_1,r_2,...,r_i) = 0$ that constrain the values of the parameters $r_1,r_2,...,r_i$. Then iterative techniques are applied to search for a set of values for the parameters that satisfies these equations, i.e., where the error functions are all zero. In practice, no such set of values exists because of machine arithmetic and imprecise or incorrect data. Instead, a set is found that minimizes the "total error", as measured by some metric, such as the sum of squares of the error functions. Using an iterative method requires the assumptions that the method terminates if no values for the parameters can give a function that matches the data tuples, that appropriate starting values for the parameters can be selected, and that an appropriate termination condition can be specified. If several numerical-analysis techniques are to be used, they should be encapsulated into a single procedure. Function Match_Form in Algorithm DATAX simply calls the procedure associated with the specified function form.

The second issue relevant to finding the constant function is the choice of the function forms. The four function forms given in Section 2.3 for P_2 illustrate that ordering for efficiency is also relevant because the first two are special cases of the last two. Choosing an ordering for the function forms corresponds to structuring the hypothesis space in what is hoped to be an efficient search order. Recent work on statistical expert systems [Phelps 87] has attempted to perform the analogous task for statistical inference. Ordering function forms in DATAX is considered in conjunction with choosing function forms because the method proposed for choosing them also yields suggestions for ordering them.

It is difficult to specify a suitable set of function forms. Any set of function forms represents a trade-off between completeness and efficiency: the larger the set of function forms, the greater the probability that it includes the right one for the data, but the longer the search time. A reasonable compromise might be a parameter which specifies the level of thoroughness of search; then the number of function forms could be selected to reflect the user's patience. In [Wu 88], 4 function forms are given, in [Wu and Wang 89], an expanded set of 20 function forms is given, and even more will be needed to apply the method to a wide range of problems. For efficiency, many function forms having closed-form solutions should be included, since they are relatively cheap to evaluate. The common models used in statistical inference should also be included. As well, it would be useful to allow the user to specify a set of function forms that must be included and a set that could be excluded, which might allow the selection of a more useful set of function forms for a particular application.

To reduce the number of function forms, the following rules are provided. A function form p_s that is a special case of a more general function form p_g should only be included if a more efficient method is available for matching against p_s than for p_g, where the efficiencies are determined by classes of time complexity, such as $O(n^2)$ versus $O(n^3)$, or by timing the matching procedures while DATAX is being used. If both function forms appear, p_s should appear before p_g in the ordering of function forms. If a number of special cases of p_g could be included in this way, then the total cost of the special cases should be compared to the cost of the p_g. If the cost of p_g is less than half of that of all the special cases combined, then only p_g should be included.

3.3 Updating Variables

In DATAX, the set of variables S is updated by replacing a subset of variables by a single variable once a constant-valued function on these variables has been discovered. In other induction programs, variables are not removed because synthetic variables are defined that *may* lead to the discovery of a constant-valued function or may be useless, rather than after a constant-valued function has been found as in DATAX. In BACON, synthetic variables (called terms) are added but existing variables are left in place, which leads to a variable set that expands without bound. In ABACUS, existing variables are left in place when synthetic variables are added, but dependencies among the variables are recorded and no effort is made to keep one variable constant if one of its dependent variables is being varied. In both of these cases, no consideration is given to the possibility that the function can be decomposed into constant-valued functions using fewer variables. When DATAX finds a constant-valued function on i variables, where i is small, it is more efficient that BACON and ABACUS because it can reduce the size of its set of variables. The crucial assumption made with the DATAX method is that there is a function form in P that will allow a constant-valued function to be found.

4 DATAX-2: A Polynomial Algorithm

To obtain a polynomial-time algorithm, a suitable inductive bias must be found. In [Utgoff 86], an *inductive bias* is defined as the "set of all factors that collectively influence hypothesis selection" (p.5). Typically, the inductive bias is determined by a combination of the hypothesis space and the method used for induction. Our approach is to define a restricted hypothesis space and then use an exhaustive search of this hypothesis space as the method.

The class of binary-partition decomposable functions, which is defined in this section, provides a suitable hypothesis space because it includes many of the numerical functions considered in the machine-discovery literature, including all those attempted by BACON.1, BACON.3 and Reduction. The class provides a tree-structured bias [Russell 88], since each hypothesis can be viewed as a tree, as will be described shortly. Algorithm DATAX-2, the polynomial-time version of DATAX presented in this section, searches the hypothesis space and performs a bottom-up construction of a tree-structured function. Following the presentation of DATAX-2, the time complexity is discussed and suggestions for improving the efficiency of the method in practice are given.

4.1 Binary-Partition Decomposable Functions

For the polynomial algorithm, the target functions are restricted to those which can be recursively decomposed into binary functions with each variable appearing in only one binary function. However, this assumption is not always correct. For example, $f(x,y,z)=xy+xz+yz$ cannot be decomposed into such binary functions; i.e., there are no functions g and h such that $f(x,y,z) = g(v_i,h(v_j,v_k))$ for any assignment of x, y, and z to one each of v_i, v_j, and v_k. These cases are handled by DATAX.

A *binary decomposable* function is defined as follows:

Definition: A function $f(V_1,...,V_n)$ is *binary decomposable* if $n \leq 2$ (such a function is called a *primitive binary function*), or if $n > 2$ and there exist functions f', g, and h such that

$f(V_1,...,V_n) = f'(g(V_a,...,V_b), h(V_c,...,V_d))$;
$\{V_a,...,V_b\} \neq \emptyset$; $\{V_c,...,V_d\} \neq \emptyset$;
$\{V_a,...,V_b\} \cup \{V_c,...,V_d\} = \{V_1,...,V_n\}$; and
$g(V_a,...,V_b)$ and $h(V_c,...,V_d)$
are both binary decomposable.

(The function f' used in the definition is called the *composition function*.)

Definition: A function is *binary-partition decomposable* if it is binary decomposable and $\{V_a,...,V_b\} \cap \{V_c,...,V_d\} = \emptyset$ at each step of the decomposition.

In other words, a binary-partition decomposable function can be decomposed into functions involving at most two variables by partitioning the variables into two nonempty subsets at each step of the decomposition. Such a decomposition can be pictured as a tree with the target function at the root, the constant-valued functions (and thus the synthetic variables) at the interior nodes, and the original variables at the leaves. For DATAX-2, the hypothesis space is restricted to the class of target equations whose target functions are binary-partition decomposable according to the definition just given.

The function forms are relevant to both the primitive binary functions and the composition functions. The set of target functions is restricted to binary-partition decomposable functions where both the primitive binary functions and the composition functions are instantiated versions of the function forms.

Binary-partition decomposable functions include the common models used in statistical inference. For example, multiple linear regression looks for equations of the form

$$\Sigma_{i=0}^{n} a_i x_i$$

where the x_i's are variables and the a_i's are associated constants; the target functions involved are binary-partition decomposable functions. Variations of this form with extra variables such as x_i^2 or x_i^3 are also tried when statistical inference is attempted in practice. Again, the relevant functions are binary-partition decomposable.

4.2 Algorithm DATAX-2

Algorithm DATAX-2(V, P, Get_Data, #D).

```
            $c \leftarrow 0$
            $S \leftarrow V$
            for $i = 2,...,2n-1$
j-loop:         for $j = 1,...,i-1$
                    if $V_j \in S$
                    then $U \leftarrow \{V_i, V_j\}$
                         $D \leftarrow$ Get_Data($V, S - U$, #$D$)
                         for all $p \in P$
                             $\{f,e\} \leftarrow$ Match_Form($p, U, D$)
                             if $f \neq$ null
                             then $c \leftarrow c + 1$
                                  Make_Variable($V_{n+c}, f, U$)
                                  $S \leftarrow S - U + \{V_{n+c}\}$
                                  exitloop(j-loop)
            if $|S| = 1$
            then return(e)
            else return(null)
```

P in this algorithm corresponds to P_2 in Algorithm DATAX. Ignoring the cost of Function Match_Form, the time complexity of DATAX-2 is $O(|P|n^2)$. The cost of Function Match_Form depends on the efficiency of the included numerical-analysis functions. A typical numerical-analysis method is Newton's method, which has quadratic convergence if a satisfactory initial approximation is provided ([Burden *et al.* 78], p.445). The method uses $O(k^3)$ operations at each step of the convergence, where k is the number of parameters in the function form. The value of k depends on the function forms included in P, but is independent of n and $|P|$.

4.3 Improving the Efficiency of DATAX-2

Since the three components of the cost of Algorithm DATAX-2 are the number of pairs of variables considered, the number of function forms, and the cost of matching with each function form, any of these components might be addressed to improve the efficiency of the algorithm. Reducing the cost of matching is outside the scope of this paper, and reducing the number of function forms has already been considered in Section 3. Two techniques are now proposed for ordering the pairs and function forms which may reduce the cost in practice.

A heuristic can be used to suggest promising pairs of variables for consideration. Typically, when checking that a function defined on a pair of variables is constant-valued, the other variables are varied one at a time. When an attempt to find a constant-valued function on two variables V_a and V_b fails when a third variable V_c is varied, the possibility is suggested that a relation may exist between V_c one of V_a and V_b. Therefore a counter for the two pairs (V_a,V_c) and (V_b,V_c) is incremented. Then, when choosing a pair of variables to examine next, the pair with the highest counter is tried first. This heuristic uses previously discovered information about possible relations to guide the search for exact descriptions of the relations. The heuristic will not affect the algorithm's ability to find a function, because it merely reorders the order in which possibilities are considered.

The function forms could be ordered according to computational cost c or according to the estimated probability p of success. By analyzing the relative frequency with which the function forms match real sets of data in a domain, p could be updated. If values were available for both c and p, the next function form could be chosen according to some function of c and p. Also, the set of function forms might be structured to allow classes of forms to be included and excluded easily and to allow results obtained when using one form to guide the choice for the next form to be considered. Of course, by excluding a class of function forms, the user might prevent the algorithm from succeeding, if the a function derived from this class was needed in composing the target function.

5 Discussion

In this paper, the DATAX method for numerical-data analysis problems has been presented. Typically, statistical analysis methods of solving these problems attempt to fit one equation model to a set of data. Numerical-analysis techniques can be applied to select one of a family of equations by approximating the parameters in an equation form. The aim of DATAX is broader still: to co-ordinate the search among many equation forms.

The function-finding problem has been characterized in Section 2 as attempting to find a nontrivial function on all variables that yields a constant value for all data tuples. Algorithm DATAX solves such problems; it can be seen as a framework because it allows the inclusion of any set of function forms and matching procedures and because as an exponential-time algorithm it clearly requires the inclusion of some heuristics based on domain knowledge for practical use.

DATAX-2 is an efficient method for discovering regularities in data if the target function can be assumed to be a binary-partition decomposable function. DATAX-2 is more efficient than Reduction [Wu 88], but is applicable to the same class of problems. Wu's Basic Reduction method begins with a set of n variables. During each of $n-1$ iterations, matches are attempted between all pairs of variables and all forms until a match is found. Then the pair of variables is replaced by a single synthetic variable and the process is repeated. Using this approach, the same match may be attempted between a form and a pair of variables on each of the n iterations, which is wasted computation because the matching results do not change. As a result, Basic Reduction requires $O(n^3)$ time in the worst case, while DATAX-2, which attempts a match between a form and a pair of variables only once, requires $O(n^2)$ time. A more complete appraisal of Wu's Reduction method is given in [Hamilton 89]. Like Basic Reduction, DATAX-2 is more efficient than ABACUS and BACON, but it achieves this efficiency by limiting the hypothesis space to binary-partition decomposable functions. According to published reports, all of the numerical functions found by ABACUS, BACON.1, and BACON.3 are binary-partition decomposable functions. However, BACON.4 has been applied to Black's heat law ([Langley *et al.* 87], p.145), which does not correspond to a binary-partition decomposable function.

Restricting attention to binary-partition decomposable functions represents applying a tree-structured bias to the set of functions. In [Russell 88], a tree-structured bias for the task of inducing predicates from Boolean variables is discussed. Russell says that a polynomial-time algorithm "has been found for the case in which the functions at each internal node of the tree are restricted to be monotone" (p.645), but he does not present the algorithm or explain the term "monotone" further. DATAX-2 is a polynomial-time algorithm in which the functions at the internal node are restricted to be binary functions and in which each original variable appears at one leaf. Since attention is restricted to Boolean functions in [Russell 88], only the 16 possible binary functions need to be considered when defining a function on 2 variables, but with DATAX-2, any binary numerical function might be applicable. Therefore, DATAX-2 performs heuristically when choosing a binary function by considering only functions corresponding to a specified set of parameterized function forms.

The next step for the development of DATAX is to incorporate a way of handling noise in data. The inability to handle noise has been identified in [Schaffer 89b] as a key weakness of current machine-discovery approaches to the function-finding problem. The first step is to specify, for each variable, the type and extent of the noise in the data values for that variable. This information could be passed by parameter to the matching routines. After a function has been chosen by the numerical-analysis procedure, we must check whether the difference between the data and this function is consistent, at some level of statistical confidence, with the specified amount of noise in the variables. This approach should supply an improvement on BACON, which has been criticized for depending on programmer-selected error bounds to obtain the desired answer [Lubinsky 89; Schaffer 89a].

DATAX could also be improved by extending it to cases where controlled presentation of data tuples is not available. Furthermore, a way of incorporating statistical inference techniques efficiently into the DATAX method is needed. DATAX-2 is less efficient than standard statistical programs because the variables are considered two at a time, rather than all at once using matrix methods.

To be applied to the search for multiple equalities in the numerical-data analysis problem, DATAX could be modified to return all constant-valued functions found, instead of null, in case of failure. However, it would only find constant-valued functions on disjoint subsets of the variables, just as ABACUS does. A major addition to DATAX would be required to find simultaneous equations that share some variables.

Acknowledgements

This paper benefited from discussions with Romas Aleliunas, Nick Cercone, Jim Delgrande, Michael Dyck, and Wilf Rosenbaum and from the comments of the anonymous CSCSI-90 referees.

References

[Burden *et al.* 78] Burden, R.L., Faires, J.D., and Reynolds, A.C. *Numerical Analysis.* Prindle, Weber, and Schmidt, Boston, MA, 1978.

[Falkenhainer and Michalski 86] Falkenhainer, B.C. and Michalski, R.S. Integrating Quantitative and Qualitative Discovery: The ABACUS System. *Machine Learning* 1(4):367-401, 1986.

[Hamilton 89] Hamilton, H.J. *Reduction Reconsidered: Further Investigations of a Mechanism for Searching for Regularity in Data.* Technical Report CSS-IS TR 89-09, Centre for Systems Science, Simon Fraser University,Burnaby, B.C., Canada V5A 1S6, December, 1989.

[Kokar 86] Kokar, M. Determining Arguments of Invariant Functional Description. *Machine Learning* 1:403-422, 1986.

[Kokar 88] Kokar, M. Syntactic Equivalence in Concept Discovery. In *Proc. of the First International Workshop in Change of Representation and Inductive Bias*, pages 268-278. 1988.

[Langley 78] Langley, P. BACON.1: A General Discovery System. In *Proc. CSCSI-78*, pages 173-180. Toronto, Ont., 1978.

[Langley *et al.* 87] Langley, P., Simon, H.A., Bradshaw, G.L. and Zytkow, J.M, *Scientific Discovery: Computational Explorations of the Creative Processes.* MIT Press, Cambridge, MA, 1987.

[Lubinsky 89] Lubinsky, D.J. Discovery from Databases: A Review of AI and Statistical Techniques. In Piatetsky-Shapiro, G. and Frawley, W. (editor), *Knowledge Discovery in Databases: IJCAI-89 Workshop Proceedings*, pages 204-218. Detroit, MI, August, 1989.

[Phelps 87] Phelps, B. (editor). *Interactions in Artificial Intelligence and Statistical Methods.* Gower Technical Press, Gower House, Croft Road, Aldershot, Hants GU11 3HR, England, 1987.

[Russell 88] Russell, S.J. Tree-Structured Bias. In *Proc. AAAI-88*, pages 641-645. St. Paul, MN, 1988.

[Schaffer 89a] Schaffer, C. An Envrionment/Classification Scheme for Evaluation of Domain-Independent Function-Finding Programs. In Piatetsky-Shapiro, G. and Frawley, W. (editor), *Knowledge Discovery in Databases: IJCAI-89 Workshop Proceedings*, pages 281-290. Detroit, MI, August, 1989.

[Schaffer 89b] Schaffer, C. BACON, Data Analysis and Artificial Intelligence. In *Proc. of the Sixth International Workshop on Machine Learning*, pages 174-178. Ithaca, NY, June, 1989.

[Simon and Lea 74] Simon, H.A. and Lea, G. Problem Solving and Rule Induction: A Unified View. In Gregg, L.W. (editors), *Models of Thought.* Yale University Press, New Haven, 1974.

[Utgoff 86] Utgoff, P. *Machine Learning of Inductive Bias.* Kluwer Academic Publishers, Norwell, MA, 1986.

[Wu 88] Wu, Y.-H. Reduction: A Practical Mechanism of Searching for Regularity in Data. In John Laird (editor), *Proc. of the Fifth International Conference on Machine Learning*, pages 374-380. University of Michigan, Ann Arbor, June, 1988.

[Wu and Wang 89] Wu, Y.-H. and Wang S. Discovering Knowledge from Observational Data. In Piatetsky-Shapiro, G. and Frawley, W. (editor), *Knowledge Discovery in Databases: IJCAI-89 Workshop Proceedings*, pages 369-377. Detroit, MI, August, 1989.

New Applications of a Fast Propositional Calculus Decision Procedure*

Shie-Jue Lee and **David A. Plaisted**
Department of Computer Science
University of North Carolina
Chapel Hill, North Carolina 27599-3175
U. S. A.

Abstract

In this paper, we present algorithms for making a fast model theoretic propositional prover informative. The prover outputs a model if the input formula is satisfiable or a minimum subset of unsatisfiable clauses if the input formula is unsatisfiable. The correctness of the result can then be verified by examining these output information. The model found by the prover also enables a new technique for problem solving in first-order logic.

1 Introduction

Deciding that a given propositional logic formula in conjunctive normal form (CNF) is satisfiable or unsatisfiable is called the satisfiability problem. It is well known that the model theoretic decision procedures can be more efficient than the proof theoretic decision procedures in solving such problems [Plaisted, 1989]. However, it is sometimes difficult for the model theoretic decision procedures to provide information for users easily to verify the result obtained.

We have already presented an intelligent propositional calculus prover in [Lee and Plaisted, 1989b]. It is a fast model theoretic decision procedure. The prover basically consists of two rules: the transitive one-literal rule and the intelligent case analysis rule. It avoids performing both cases in a case analysis all the time by checking the proof dependency information. It has been shown to run faster than the Davis-Putnam method in most cases. However, it also suffers the disadvantage of providing no information about the result for users.

In this paper, we present algorithms that can be implemented in the intelligent propositional calculus prover to make it informative without much overhead. A model is provided if the input formula is satisfiable, while a minimum subset of unsatisfiable clauses is found if the input foumula is unsatisfiable. The correctness of the result can then be verified by examining this output information. An easy, short program may be written for verifying if the input formula is indeed satisfied by the output model. Usually, the number of clauses in a minimum subset of unsatisfiable clauses for an input formula is small. Therefore, a simple, constructive yet less efficient proof theoretic decision procedure with tracing capability may be used to verify if the subset is indeed unsatisfiable. Most of the time, the verification can be done by hand very easily.

The model found by the propositional calculus prover also provides a new technique for problem solving in first-order logic. The propositional calculus prover works in conjunction with an instance based first-order theorem prover, IBTP, for this purpose. IBTP generates instances of the axioms of a given problem. Then the propositional calculus prover is used to find a model for the instances. A ground literal in the model is a possible logical consequence of the axioms. This can be verified by running IBTP on the conjunction of the axioms and the negation of the ground literal.

Throughout this paper, a disjunction of literals may also be represented as a set of literals, called a *clause*. A formula in conjunctive normal form may also be represented as a set of clauses. A propositional constant p is called a *positive literal*, while $\neg p$ is called a *negative literal*. A clause with all positive literals is called a *positive clause*. A clause with all negative literals is called a *negative clause*.

In the sequel, we begin with an introduction of the intelligent propositional calculus prover. Then we present the algorithms for making the prover informative. Finally, we present the new technique for problem solving in first-order logic enabled by the capability of finding models of the propositional calculus prover.

2 The Intelligent Propositional Calculus Prover

Definition. Suppose F is a set of clauses and contains a unit clause $\{p\}$. Deleting all clauses containing p from F is called *(propositional) unit subsumption.* ♡

Definition. Suppose F is a set of clauses and contains a unit clause $\{p\}$. Replacing C by $C - \{\neg p\}$ in F for all clauses C containing $\neg p$ is called *(propositional) unit simplification.* ♡

Definition. Suppose a set F of clauses is of the form

$$(A_1 \vee p) \wedge \cdots \wedge (A_m \vee p) \wedge (B_1 \vee \neg p) \wedge \cdots \wedge (B_n \vee \neg p) \wedge R$$

*This research was supported by NSF under grant CCR-8802282.

where A_i, $1 \leq i \leq m$, B_j, $1 \leq j \leq n$, and R are free of p and $\neg p$. Let F_1 be the set

$$B_1 \wedge \cdots \wedge B_n \wedge R$$

and F_2 be the set

$$A_1 \wedge \cdots \wedge A_m \wedge R$$

F_1 is called the ***affirmative case***. p is called the ***affirmative assumption***. The simplification of clauses $B_1 \vee \neg p$, ..., $B_n \vee \neg p$ is called *assumption simplification* with assumption p. F_2 is called the ***negative case***. $\neg p$ is called the ***negative assumption***. The simplification of clauses $A_1 \vee p$, ..., $A_m \vee p$ is called assumption simplification with assumption $\neg p$. ♡

Definition. An assumption p is *relevant* to a clause C if one of the following conditions holds:

1. C is a clause produced by assumption simplification with assumption p.
2. If p is relevant to a unit clause U, then p is also relevant to all the clauses simplified by unit simplification with U.

We call p a *relevant assumption* to C. We also say that C ***depends*** on p. ♡

The intelligent propositional calculus prover consists of two rules: the transitive one-literal rule and the intelligent case-analysis rule.

1. The transitive one-literal rule. Suppose F is a set of clauses.
   (a) It performs unit subsumption.
   (b) It performs unit simplification.
   (c) It passes all the relevant assumptions of a unit clause to all the clauses simplified by the unit clause during unit simplification.
   (d) During unit simplification, if a simplified clause is the empty clause, then F is unsatisfiable and the unsatisfiability of F depends on all the relevant assumptions of this empty clause.
2. The intelligent case-analysis rule. Suppose F is a set of clauses.
   (a) If F does not contain positive clauses or negative clauses, then F is satisfiable.
   (b) If F contains an empty clause, then F is unsatisfiable and the unsatisfiability of F depends on all the relevant assumptions of this empty clause.
   (c) Otherwise, pick the first negative clause C from F with least number of literals. Pick a literal p from C as the affirmative assumption and compute the affirmative case F_1. All the clauses simplified by p depend on p.
      i. If F_1 is satisfiable, then F is satisfiable.
      ii. If F_1 is unsatisfiable and the unsatisfiability of F_1 depends on p, compute the negative case F_2 with the negative assumption $\neg p$. All the clauses simplified by $\neg p$ depend on $\neg p$.
         A. If F_2 is satisfiable, then F is satisfiable.
         B. If F_2 is unsatisfiable and the unsatisfiability of F_2 depends on $\neg p$, then F is unsatisfiable and the unsatisfiability of F depends on what F_1 or F_2 depends.
         C. If F_2 is unsatisfiable and the unsatisfiability of F_2 does not depend on $\neg p$, then F is unsatisfiable and the unsatisfiability of F depends on what F_2 depends.
      iii. If F_1 is unsatisfiable and the unsatisfiability of F_1 does not depend on p, then F is unsatisfiable and the unsatisfiability of F depends on what F_1 depends.

The intelligent propositional calculus prover applies the transitive one-literal rule and the intelligent case-analysis rule iteratively. The prover terminates on any given set F of clauses since the size of F is monotonically decreasing by applying these two rules.

3 Making The Prover Informative

In this section, we present algorithms that make the intelligent propositional calculus prover informative without much overhead. If a set F of input clauses is satisfiable, the prover outputs a model that makes all the clauses in F true. If F is unsatisfiable, the prover outputs a minimum subset R of unsatisfiable clauses for S, i.e., if C is a clause in R, then $R - \{C\}$ is satisfiable.

Definition. A clause C is *relevant* to a clause D if one of the following conditions holds:

1. If C is a unit clause, and D is a clause simplified by C during unit simplification.
2. If C is relevant to a unit clause U, and D is a clause simplified by U.

We call C a *relevant clause* to D, or D ***depends*** on C. If C is an input clause, then C is also a relevant clause to itself. ♡

In the following algorithms, procedure IP is the top level call. It returns a model M if the input set F is satisfiable or a minimum subset of unsatisfiable clauses of F if F is unsatisfiable. Procedure TOLR performs transitive one-literal rule. Procedure ICAR carries out intelligent case-analysis rule. Procedure PDE computes dependencies if both cases in a case analysis are unsatisfiable.

- Input: A CNF formula or clause set F. M is set empty.
- Output: If F is satisfiable, then Tag is true and M is a model. If F is unsatisfiable, then Tag is false and R_F is a minimum subset of unsatisfiable clauses for F.

Algorithms of the informative prover

```
IP($F$,Tag,$D_F$,$M$,$R_F$)
  if $F$ is
    case 1: empty
    case2: non-positive;
        $M = M \cup \{p \in F : p \textit{ is a negative literal}\}$
    case3: non-negative;
        $M = M \cup \{p \in F : p \textit{ is a positive literal}\}$
  then Tag = true                    % satisfiable
```

```
else if $F$ contains an empty clause $E$
    then Tag = false;
        $D_F$ contains the relevant assumptions
        of $E$;
        $R_F$ contains the relevant clauses of $E$
    else TOLR($F$,Tag,$D_F$,$M$,$R_F$);
        if $F$ is empty
        then Tag = true            % satisfiable
        else if Tag = false
            then return
            else ICAR($F$,Tag,$D_F$,$M$,$R_F$)
            fi
        fi
    fi
fi
TOLR($F$,Tag,$D_F$,$M$,$R_F$)
  while $F$ has a unit clause $U = \{q\}$ do
      do unit subsumption and
      unit simplification on $F$;
      pass the relevant assumptions of $U$ to
      all the clauses simplified by $U$;
      pass the relevant clauses of $U$ to
      all the clauses simplified by $U$;
      $M = M \cup \{p\}$;
      if a simplified clause $E$ is {}
      then Tag = false;
          $D_F$ contains the relevant assumptions
          of $E$;
          $R_F$ contains the relevant clauses of $E$;
          return
      fi
      let the resulting set be $F$
  endwhile
ICAR($F$,Tag,$D_F$,$M$,$R_F$)
  pick a negative literal $p$ from
  the first smallest negative clause in $F$;
  let $F$ be
      $(A_1 \lor p) \land \cdots \land (A_m \lor p) \land$
      $(B_1 \lor \neg p) \land \cdots \land (B_n \lor \neg p) \land R$;
  let $F_1$ be
      $B_1 \land \cdots \land B_n \land R$;
  add $p$ as a relevant assumption to $B_1, \ldots, B_n$;
  $M_1 = M \cup \{p\}$;
  IP($F_1$,Tag1,$D_{F_1}$,$M_1$,$R_{F_1}$);     % affirmative case
  if Tag1                      % satisfiable
  then Tag = Tag1;
      $M = M_1$
  else if $p \in D_{F_1}$
      then let $F_2$ be
              $A_1 \land \cdots \land A_m \land R$;
          add $\neg p$ as a relevant assumption to
          $A_1, \ldots, A_m$;
          $M_2 = M \cup \{\neg p\}$;
          IP($F_2$,Tag2,$D_{F_2}$,$M_2$,$R_{F_2}$);
          % negative case
          if Tag2                  % satisfiable
          then Tag = Tag2;
              $M = M_2$
          else PDE($p$,$D_{F_1}$,$D_{F_2}$,$D_F$,$R_{F_1}$,$R_{F_2}$,$R_F$)
          fi
      else Tag = Tag1;
          $D_F = D_{F_1}$;
          $R_F = R_{F_1}$
      fi
  fi
PDE($p$,$D_{F_1}$,$D_{F_2}$,$D_F$,$R_{F_1}$,$R_{F_2}$,$R_F$)
  if $\neg p \in D_{F_2}$
  then $D_F = (D_{F_1} - \{p\}) \cup (D_{F_2} - \{\neg p\})$;
      $R_F = R_{F_1} \cup R_{F_2}$
  else $D_F = D_{F_2}$;
      $R_F = R_{F_2}$
  fi
♡
```

4 Examples

Two examples are given here to show how the output information provided by the prover look like. Predicate symbols with ground terms as arguments denote propsitional symbols.

Example 1. Suppose S contains the following clauses:

1. $\{\neg p(b,a), \neg p(b,f(b)), \neg p(f(b),b)\}$
2. $\{\neg p(a,a)\}$
3. $\{\neg p(f(b),a), \neg p(f(b),b), \neg p(b,f(b))\}$
4. $\{p(b,f(b)), p(b,a)\}$
5. $\{p(f(b),b), p(b,a)\}$
6. $\{\neg p(b,a), \neg p(b,b), \neg p(b,b)\}$
7. $\{p(f(a),a)\}$
8. $\{\neg p(f(a),f(f(a))), \neg p(f(f(a)),f(a))\}$
9. $\{\neg p(a,f(a))\}$

The prover detects S being satisfiable and outputs the following model M:

- $\neg p(f(b),a)$
- $p(f(b),b)$
- $p(b,f(b))$
- $\neg p(b,a)$
- $\neg p(f(a),f(f(a)))$
- $\neg p(a,f(a))$
- $p(f(a),a)$
- $\neg p(a,a)$

It can be examined by hand very easily that M indeed makes all the clauses in S true. A very simple program may also be used to do this check. ♡

Example 2. Suppose S contains the following clauses:

1. $\{g(f(a),a), g(a,a)\}$
2. $\{g(f(f(a)),f(a)), \neg g(a,f(a))\}$
3. $\{g(f(f(f(b))),f(f(b))), \neg g(f(b),f(f(b)))\}$
4. $\{g(f(f(b)),f(b)), \neg g(b,f(b))\}$
5. $\{g(f(b),f(f(b))), \neg g(f(f(b)),f(b))\}$
6. $\{g(b,f(b)), \neg g(f(b),b)\}$

7. $\{g(f(a), f(f(a))), \neg g(a, f(a))\}$
8. $\{g(f(f(b)), f(f(f(b)))), \neg g(f(b), f(f(b)))\}$
9. $\{g(f(b), f(f(b))), \neg g(b, f(b))\}$
10. $\{\neg g(f(a), a), \neg g(f(f(a)), f(a))\}$
11. $\{\neg g(f(a), a), \neg g(a, f(a))\}$
12. $\{\neg g(f(b), a), \neg g(f(f(b)), f(b))\}$
13. $\{\neg g(b, a), \neg g(f(b), b)\}$
14. $\{\neg g(a, a)\}$
15. $\{\neg g(f(f(b)), a), \neg g(f(b), f(f(b)))\}$
16. $\{\neg g(f(b), a), \neg g(b, f(b))\}$
17. $\{g(f(f(a)), f(a)), g(f(a), a)\}$
18. $\{g(f(f(b)), f(b)), g(f(b), a)\}$
19. $\{g(a, f(a)), g(a, a)\}$
20. $\{g(f(a), f(f(a))), g(f(a), a)\}$
21. $\{g(f(b), f(f(b))), g(f(b), a)\}$
22. $\{g(f(b), b), g(b, a)\}$
23. $\{g(b, f(b)), g(b, a)\}$
24. $\{g(f(b), b), \neg g(b, b)\}$
25. $\{g(b, f(b)), \neg g(b, b)\}$
26. $\{\neg g(b, a), \neg g(b, b)\}$

The prover detects S being unsatisfiable and outputs the following minimum subset R of unsatisfiable clauses:

1. $\{g(f(a), a), g(a, a)\}$ (clause 1)
2. $\{g(f(f(a)), f(a)), \neg g(a, f(a))\}$ (clause 2)
3. $\{\neg g(f(a), a), \neg g(f(f(a)), f(a))\}$ (clause 10)
4. $\{\neg g(a, a)\}$ (clause 14)
5. $\{g(a, f(a)), g(a, a)\}$ (clause 19)

Again, it is easy to check by hand that R is indeed minimum and unsatisfiable. We may also use a proof theoretic decision procedure [Robinson, 1965; Chang and Lee, 1973; Loveland, 1978, Plaisted, 1988; Stickel, 1988], equipped with tracing capability, to verify the unsatisfiability of R. ♡

5 Implementation and Test

We have implemented the informative propositional calculus prover in Prolog. The Prolog system we used is ALS-Prolog Compiler (Version 0.60). The following are some remarks about the implementation:

1. Prolog has no occurs check. We write a sound unification algorithm in Prolog.
2. We take advantage of the fast implementation of failure in Prolog. We don't use tags to indicate satisfiable or unsatisfiable for procedures. A procedure succeeds if the input to this procedure is unsatisfiable and fails if the input is satisfiable.
3. A flag *out_model_rc* is provided to do the following things if it is turned on:
   - recording relevant clauses for each clause
   - printing out a model before the deepest-level call fails
   - printing out a minimum subset of unsatisfiable clauses when the top-level call succeeds if the input set of clauses is unsatisfiable

   The model construction is performed regardless of the setting of the flag, since it does not slow down the prover noticeably much from the experiments we have done.
4. A technique, called delayed merging, is used for finding a minimum subset of unsatisfiable clauses. A recursive expression of relevant clauses is created for each clause until an empty clause is found, where the expression of relevant clauses with this empty clause is evaluated to form an ordered list. For example, suppose X and Y are two expressions of relevant clauses. We use merge(X,Y,Z) to denote that Z is the resulting expression of the union of X and Y, where Z is a variable and X and Y are similar expressions. We don't evaluate Z until the clause with Z is an empty clause. The evaluation of an expression is done recursively.

Adding the capability of providing information for users to the intelligent propositional calculus prover only incurs a little overhead. This is shown in Table 1[1] at the end of the paper. In Table 1, the number of clauses and propositional constants for each problems are listed in third and fourth columns respectively. 'Sat' in fifth column indicates that the problem is satisfiable, while 'Unsat' indicates that the problem is unsatisfiable. The running times for each problem are listed in columns 6-8. Column 6 is for the intelligent propositional prover without the capability of constructing and printing out information. Columns 7 and 8 are for the informative intelligent propositional prover with *out_model_rc* turned off and *out_model_rc* turned on respectively.

The test problems we used are the ground instance sets of first-order logic problems. These ground instance sets are generated by an instance based theorem prover IBTP in our research. Problems 1–36 come from the problem set of [Stickel, 1988]. The problem "example" is a theorem presented by Pellitier and Rudnicki in AAR Newsletter No. 6, 1986. The problem "salt" is the salt and mustard problem. Problems 41–43 are pigeon hole theorems; they are propositional problems. Problems 44-46 are obtained from propositional temporal logic theorems.

6 A New Technique For Problem Solving

6.1 Descriptions

The model finding capability of the propositional calculus prover provides a new technique for solving problems of first-order logic. The prover works in conjunction with IBTP for this purpose. IBTP is an instance based theorem prover developed for our theorem proving research.

[1]The times are obtained on a SUN3/60 workstation with 12 MB memery

It is complete for first-order logic, namely, given any theorem it will eventually find a proof. It basically applies a hyper-matching strategy [Lee and Plaisted, 1989a] to generate instances of the input clauses, and then tries to find a contradiction from the instances obtained so far. If a contradiction is found, then the input clauses are unsatisfiable and we are done. Otherwise, more instances are generated.

A typical problem for a theorem prover to solve consists of a set of axioms and the conclusion to be proved. However, in some cases, we may only have a set of axioms described as rules or constraints. We don't know what conclusions can be drawn from these axioms. Typical examples are puzzles. Therefore, it is hard to get help from a normal theorem prover in these cases.

A new technique for solving such kind of problems is basically as follows. Suppose we have a set A of axioms. We use IBTP to generate the instances of A, then call the informative propositional calculus prover. Since A is satisfiable, a model M can be found for these instances. Suppose M contains a literal p. Then p is a possible logical consequence of A. To verify that p is a conclusion of A, we can run IBTP again on the set $A \cup \{\neg p\}$.

This technique works well, especially for those problems whose Herbrand universe is finite. Many problems, i.e. puzzles, have finite Herbrand universe. If problems have a certain form (no function symbols), then the Herbrand universe is guaranteed to be finite. In this case, IBTP works as a decision procedure; it will stop with an answer. For such problems, we can let IBTP generate all instances of the axioms with hyper-matching strategy before calling the informative propositional calculus prover. We may find out all the possible conclusions we want from the model obtained by the propositional calculus prover.

6.2 Examples

Three examples are given here for illustrations.

Example 3. Here is a description for a logic puzzle "friends" [Sterling and Shapiro, 1986]:

1. There are three friends: Michael, Richard, and Simon; three nationalities: American, Australian, and Israeli; three sports: basketball, cricket, and tennis. Each friend has a unique nationality and plays a unique sport.
2. These friends came first, second, and third in a programming competition.
3. Michael likes basketball, and did better than the American.
4. Simon, the Israeli, did better than the tennis player.
5. The cricket player came first.

These rules can be expressed in first order logic as follows:

1. $\forall P \exists N$ nationality(P, N)
   (each person has a nationality)
2. $\forall P_1, N_1, P_2, N_2$ (nationality$(P_1, N_1) \wedge$ nationality$(P_2, N_2) \wedge P_1 \neq P_2 \rightarrow N_1 \neq N_2)$
   (uniqueness of nationality)
3. $\forall P \exists S$ play(P, S)
   (each person plays a sport)
4. $\forall P_1, S_1, P_2, S_2$ (play$(P_1, S_1) \wedge$ play$(P_2, S_2) \wedge P_1 \neq P_2 \rightarrow S_1 \neq S_2)$
   (uniqueness of sport)
5. $\forall P \exists O$ came(P, O)
   (each person came in an order)
6. $\forall P_1, O_1, P_2, O_2$ (came$(P_1, O_1) \wedge$ came$(P_2, O_2) \wedge P_1 \neq P_2 \rightarrow O_1 \neq O_2)$
   (uniqueness of order)
7. play(michael,basketball)
8. $\forall P$ (nationality(P,american) $\rightarrow$ did_better(michael,P))
9. nationality(simon,israeli)
10. $\forall P$ (play(P,tennis) $\rightarrow$ did_better(simon,P))
11. $\forall P_1, P_2$ (did_better$(P_1, P_2) \vee$ did_better$(P_2, P_1))$
12. $\forall P_1, P_2$ (did_better$(P_1, P_2) \rightarrow \neg$did_better$(P_2, P_1))$
    (exclusiveness of did_better)
13. $\forall P_1, P_2, P_3$ (did_better$(P_1, P_2) \wedge$ did_better$(P_2, P_3) \rightarrow$ did_better$(P_1, P_3))$
    (transitivity of did_better)
14. $\forall P$ (play(P,cricket) $\rightarrow$ came(P,1))

We transformed these first-order formulas into function-free first-order formulas using the transformation rules that will be described later. The function-free first-order formulas are then converted into a set A of clauses by a standard procedure [Chang and Lee, 1973; Loveland, 1978]. Then we use IBTP to generate all the instances of A with hyper-matching strategy. The informative propositional calculus prover is called to find a model M for these instances. The following literals are contained in M:

- came(simon,1)
- nationality(michael,australian)
- nationality(richard,american)
- nationality(simon,israeli)
- play(michael,basketball)
- play(richard,tennis)
- play(simon,cricket),
- did_better(michael,richard)
- did_better(simon,richard)

These literals are indeed logical consequences of the above rules. This can be verified by running IBTP on A and the disjunction of all the negated literals above. The total time for solving this problem is 29.783[2] seconds and 0.300 seconds of it is spent by the informative propositional calculus prover to find the model. ♡

Example 4. The following constraints are for the "jobs" puzzle found in [Wos *et al.*, 1984]:

[2]All the times in this section are obtained on a SUN3/60 workstation with 12 MB memery

1. There are four people: Roberta, Thelma, Pete, and Steve; eight jobs: actor, boxer, chef, guard, nurse, police officer, teacher, and telephone operator.
2. Each person holds exactly two jobs.
3. The job of nurse is held by a male.
4. The husband of the chef is the telephone operator.
5. Roberta is not a boxer.
6. Pete is not college educated.
7. Roberta, chef, and the police officer went golfing together.
8. The nurse, teacher, and police officer are all college educated.

A model found for the above constraints contains the following literals:

- job(roberta,guard)
- job(roberta,teacher),
- job(thelma,boxer)
- job(thelma,chef),
- job(pete,actor)
- job(pete,telephone_operator),
- job(steve,nurse)
- job(steve,police_officer),
- husband(pete,thelma)

These literals are indeed logical consequences of the constraints. The total time for solving this problem is 351.683 seconds and 46.183 seconds of it is spent by the informative propositional calculus prover to find the model. ♡

Example 5. The following rules are for a problem called "zebra" [Sterling and Shapiro, 1986]:

1. There are five people: englishman, spaniard, norwegian, japanese, and ukranian; five houses: 1, 2, 3, 4, and 5. five drinks: orange, coffee, water, tea, and milk; five cigarettes: gold, kools, chestfields, lucky, and parliaments; five animals: dog, fox, horse, snails, and zebra; five colors: yellow, red, blue, ivory, and green. Each people lives in a unique house, drinks a unique drink, owns a unique animal, smokes a unique cigarette, and each house has a unique color.
2. The englishman lives in the red house.
3. The spaniard owns a dog.
4. The norwegian lives in the first house.
5. Kools are smoked in the yellow house.
6. Chesterfields are smoked next to where the fox is kept.
7. The norwegian lives next to the blue house.
8. The gold smoker owns snails.
9. The lucky smoker drinks orange juice.
10. The ukranian drinks tea.
11. The japanese smokes parliaments.
12. The kools smoker lives next to where the horse horse is kept.
13. Coffee is drunk in the green house.
14. The green house is to the immediate right of the ivory house.
15. Milk is drunk in the middle house.

A model found for the above rules contains contains the following literals:

- color(1,yellow)
- color(2,blue)
- color(3,red)
- color(4,ivory)
- color(5,green)
- lives(norwegian,1)
- lives(ukranian,2)
- lives(english,3)
- lives(spaniard,4)
- lives(japanese,5)
- owns(norwegian,fox)
- owns(ukranian,horse)
- owns(english,snails)
- owns(spaniard,dog)
- owns(japanese,zebra)
- smokes(norwegian,kools)
- smokes(ukranian,chestfields)
- smokes(english,gold)
- smokes(spaniard,lucky)
- smokes(japanese,parliaments)
- drinks(norwegian,water)
- drinks(ukranian,tea)
- drinks(english,milk)
- drinks(spaniard,orange)
- drinks(japanese,coffee)

These literals are indeed logical consequences of the rules. The total time for solving this problem is 1805.950 seconds and 157.033 seconds of it is spent by the informative propositional calculus prover to find the model. ♡

Some problems, like 8 queens, have more than one solution. Our prover can be used to find all the solutions, one by one, by finding models and negating them and adding to the set of axioms.

One point we should mention here. Our axiomatizations all have the characteristic that they are expressed in function free quantifier free first order logic, which is decidable since the Herbrand universe is finite. Also, these axiomatizations are fully declarative, and do not use negation as failure as Prolog does. Therefore, these formulations are more natural than Prolog ones, even though the times to solve the problems are often slower than Prolog times.

7 Transformation rules

In this section, we describe some rules of transforming a first-order formula with finite domain into function-free first-order formulas. The function-free first-order formulas guarantee the finiteness of the Herbrand universe, thus our prover is guaranteed to stop with an answer.

Suppose the domain of the problem contains elements a_1, ..., and a_n.

1. Suppose there is an expression:
$$\forall X \exists Y p(X, Y)$$
The expression can be transformed into the following function-free expression:
$$\forall X (p(X, a_1) \vee p(X, a_2) \vee \cdots \vee p(X, a_n))$$
2. Consider the following expression:
$$\exists X \forall Y p(X, Y)$$
This expression can be transformed into the following function-free expression:
$$(\forall Y_1 p(a_1, Y_1)) \vee (\forall Y_2 p(a_2, Y_2)) \vee \cdots \vee (\forall Y_n p(a_n, Y_n))$$
3. The expression for uniqueness often has the following form:
$$(\forall X_1 X_2 Y_1 Y_2 (p(X_1, Y_1) \wedge p(X_2, Y_2) \wedge X_1 \neq X_2 \rightarrow Y_1 \neq Y_2)$$
we may replace it by the following set of formulas:
$$\begin{gathered} \forall X \neg p(a_1, X) \vee \neg p(a_2, X) \\ \forall X \neg p(a_1, X) \vee \neg p(a_3, X) \\ \vdots \\ \forall X \neg p(a_1, X) \vee \neg p(a_n, X) \\ \forall X \neg p(a_2, X) \vee \neg p(a_3, X) \\ \vdots \\ \forall X \neg p(a_2, X) \vee \neg p(a_n, X) \\ \vdots \\ \forall X \neg p(a_{n-1}) \vee \neg p(a_n, X) \end{gathered}$$

Note that the transformation can be done mechanically.

8 Conclusion

We have briefly introduced the idea of an intelligent propositional calculus prover, which is a fast model theoretic decision procedure. We have also presented algorithms for making the prover informative without much overhead. A model is provided if the input formula is satisfiable, while a minimum subset of unsatisfiable clauses is found if the input foumula is unsatisfiable.

The output information can be used for verifying the result obtained by the prover. An easy, short program may be written for verifying if the input formula is indeed satisfied by the output model. Usually, the number of clauses in a minimum subset of unsatisfiable clauses for an input formula is small. Therefore, a simple, constructive yet less efficient proof theoretic decision procedure with tracing capability may be used to verify if the subset is indeed unsatisfiable. Most of the time, the verification can be done by hand very easily.

The model found by the propositional calculus prover also provides a new technique for solving problems of first-order logic. The propositional calculus prover works in conjunction with an instance based first-order theorem prover, IBTP, for this purpose. IBTP generates instances of the axioms of a given problem. Then the propositional calculus prover is used to find a model for these instances. A ground literal in the model is a possible logical consequence of the axioms. This can be verified by running IBTP on the conjunction of the axioms and the negation of the ground literal.

Acknowledgements

We thank the referees for the comments.

References

[Chang and Lee, 1973] C. Chang and and R. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, New York, 1973.

[Lee and Plaisted, 1989a] S.J. Lee and D. Plaisted. Theorem Proving using Hyper-Matching Strategy. *Proc. Fourth International Symp. on Methodologies for Intelligent Systems,* 467-476, 1989.

[Lee and Plaisted, 1989b] S.J. Lee and D. Plaisted. An Intelligent Decision Procedure for Propositional Logic. Department of Computer Science, The University of North Carolina at Chapel Hill, 1989.

[Loveland, 1978] D. Loveland. *Automated Theorem Proving: A Logical Basis.* North-Holland, New York, 1978.

[Plaisted, 1988] D. Plaisted. Non-Horn Clause Logic Programming Without Contrapositives. *Journal of Automated Reasoning,* 4(3):287-325, 1988.

[Plaisted, 1989] D. Plaisted. Mechanical Theorem Proving. In *A Sourcebook on Formal Techniques in Artificial Intelligence,* R. Banerji (ed.). Elsevier, Amsterdam, 1989.

[Robinson, 1965] J. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *J. ACM,* 12(1):23-41, 1965.

[Sterling and Shapiro, 1986] L. Sterling and E. Shapiro. *The Art of Prolog.* The MIT Press, Massachusetts, 1986.

[Stickel, 1988] M.E. Stickel. A Prolog Technoligy Theorem Prover: Implementation by an Extended Prolog Compiler. *Journal of Automated Reasoning,* 4(4):353-380, 1988.

[Wos *et al.*, 1984] L. Wos, R. Overbeek, E. Lusk, and J. Boyle. *Automated Reasoning: Introduction and Applications.* Prentice-Hall, 1984.

Table 1. Sample running times[3]

No	problem	no. of input clauses	no. of prop. consts	satisfi-ability	PC	INF-PC with flag off	INF-PC with flag on
1	burstall	36	37	Sat	0.150	0.133	0.233
2	shortburst	15	16	Sat	0.017	0.017	0.067
3	prim	31	40	Sat	0.117	0.117	0.183
4	hasparts2	33	49	Unsat	0.083	0.083	0.100
5	group2	11	10	Sat	0.017	0.033	0.067
6	ew2	5	3	Unsat	0.000	0.017	0.017
7	ew3	9	5	Unsat	0.017	0.033	0.050
8	rob2	27	34	Sat	0.067	0.083	0.117
9	qw	9	10	Sat	0.017	0.033	0.050
10	mqw	26	15	Unsat	0.017	0.033	0.033
11	dbabhp	42	50	Sat	0.183	0.167	0.217
12	ex4-t1	107	54	Unsat	0.900	0.867	1.183
13	ex4-t2	107	54	Unsat	0.900	0.917	1.100
14	ex5	498	324	Unsat	3.967	3.867	3.933
15	ex6-t1	62	58	Sat	0.367	0.350	0.500
16	ex6-t2	62	58	Sat	0.383	0.383	0.500
17	wos2	29	26	Sat	0.083	0.083	0.133
18	wos4	134	101	Unsat	0.767	0.700	0.717
19	wos6	146	138	Sat	2.167	2.117	2.683
20	wos7	104	104	Sat	1.100	1.117	1.417
21	wos9	390	387	Unsat	0.017	0.000	0.000
22	wos11	193	185	Sat	3.700	3.650	4.333
23	wos16	122	116	Sat	1.600	1.533	1.917
24	wos17	159	159	Sat	2.583	2.567	3.083
25	wos25	172	193	Sat	1.367	1.350	1.500
26	wos33	103	125	Sat	0.717	0.717	0.817
27	ls103	105	49	Unsat	0.283	0.250	0.300
28	ls106	23	17	Sat	0.050	0.050	0.100
29	ls108	288	328	Unsat	1.267	1.300	1.350
30	ls17	32	38	Sat	0.083	0.083	0.150
31	ls28	351	276	Sat	8.800	8.767	9.267
32	ls29	348	277	Unsat	1.217	1.233	1.267
33	ls5	6	4	Unsat	0.000	0.017	0.033
34	ls65	61	58	Sat	0.333	0.333	0.517
35	ls75	56	55	Sat	0.333	0.317	0.633
36	ls112	699	754	Unsat	6.333	6.233	6.333
37	example	136	47	Unsat	0.750	0.783	0.833
38	expq	4	2	Unsat	0.000	0.000	0.033
39	liar	39	35	Unsat	0.033	0.067	0.100
40	salt	104	30	Unsat	2.400	2.433	2.733
41	ph4	22	12	Unsat	0.133	0.150	0.183
42	ph5	45	20	Unsat	0.833	0.817	1.033
43	ph6	81	30	Unsat	5.000	5.067	6.400
44	tempo1	27	16	Unsat	0.000	0.033	0.017
45	tempo2	11	8	Unsat	0.017	0.017	0.033
46	tempo3	10	7	Unsat	0.017	0.000	0.017
	Average				1.070	1.070	1.223

[3]The unit of time is second. For the descriptions of the table, see section 5

On Theorem Provers for Circumscription

Katsumi Inoue and **Nicolas Helft**
ICOT
Research Center, Mita Kokusai Bldg. 21F
1-4-28 Mita, Minato-ku, Tokyo 108
Japan

Abstract

This paper concerns algorithms to answer queries in circumscriptive theories. Two recent papers present such algorithms that are relatively complex: Przymusinski's algorithm is based on MILO-resolution, a variant of ordered linear resolution; Ginsberg's theorem prover uses a backward-chaining ATMS. Because of their different concerns, formalisms, and implementation, it is not clear what their relative advantages are. This paper makes a detailed comparison of these relating them to a logical framework of abduction, explains their intuitive meaning, and shows how the efficiency of both can be improved. Additionally, some limitations of both circumscriptive theorem provers are also discussed.

1 Introduction

Circumscription [McCarthy, 1980; Lifschitz, 1985] is one of the most powerful and well-developed formalizations of nonmonotonic reasoning as it is based on classical predicate logic. Although its formal properties are well investigated, there have been few attempts at effective query answering procedures or implementations for circumscriptive theories.

Recently, Przymusinski [1989] and Ginsberg [1989] have published algorithms to compute circumscription. Ginsberg acknowledges a strong connection between the results presented. However, not much is known about the algorithms' relative advantages and disadvantages.

The goal of this paper is twofold:

1. We further explore the connections between algorithms [Przymusinski, 1989; Ginsberg, 1989], showing that:
   (a) The theoretical results obtained in each of these papers are the same, and both can be re-expressed in a simple, general framework.
   (b) The algorithms presented have different computational properties; we provide a detailed comparison of these.
2. We show how the efficiency of both algorithms can be improved.

Sections 2, 3, and 4 consider the above questions. In Section 5, we further discuss two important problems that arise in Przymusinski's and Ginsberg's approaches and suggest some solutions to them.

2 Comparing the Theorems

We will consider ground theories, that is, first-order theories, without equality, consisting of finitely many ground formulas over the representation language $\mathcal{L}$; these are sufficient to illustrate the comparison between the algorithms of [Przymusinski, 1989; Ginsberg, 1989]. We will use the clausal form representation, and also assume that Unique Names Axioms (UNA) are satisfied for $\mathcal{L}$, as in both algorithms. According to Przymusinski's claims, however, the algorithms are applicable to the first-order case with UNA and equality axioms. Ginsberg adds the domain-closure axiom, which is unnecessary according to the results we shall present which indicate the equivalence between the theoretical results of [Przymusinski, 1989; Ginsberg, 1989]. In Section 5, we will return to the incompleteness problem, which is due to the infinite properties of first-order theories.

We briefly recall a basic property of circumscription, on which the algorithms are based. The predicate symbols of a theory T are divided into three disjoint sets: P, *minimized* predicates; Z, *variables*; and Q, *fixed*. Using this information, some models of T are defined as minimal with respect to the sets P and Z; we say they are (P, Z)*-minimal*. Let $CIRC(T; P; Z)$ be the circumscription of P in T with variable predicates Z. Then, for any formula F, $CIRC(T; P; Z) \models F$ iff $M \models F$ for every (P, Z)-minimal model M of T [McCarthy, 1980; Lifschitz, 1985].

Now, to compare the theoretical results of Przymusinski and Ginsberg, we use the notion of *characteristic clauses* which was introduced by Bossu & Siegel [1985] and was later generalized by Siegel [1987]. This concept can also give the computational aspect of *abduction*. Informally speaking, characteristic clauses are intended to represent "interesting" clauses to solve a certain problem, and are constructed over a sub-vocabulary of $\mathcal{L}$ called a *production field*.

Definition 2.1 A *production field* $\mathcal{P}$ is a set of ground literals. A clause C *belongs to a production field* $\mathcal{P}$ if every literal in C belongs to $\mathcal{P}$. The set of clauses that are logical consequences of a set of clauses T and that belong to $\mathcal{P}$ is denoted by $Th_{\mathcal{P}}(T)$.

If R is a set of predicate symbols, we denote by R^+ (respectively R^-) the positive (respectively negative) ground literals with predicates from R, which range over all constants in $\mathcal{L}$. Moreover, $R^+ \cup R^-$ is denoted $R^{\pm}$.

Example 2.2 Suppose that the language $\mathcal{L}$ contains predicates, ***bird***, ***flies***, ***ab***, and ***ostrich***, and that ***tweety*** is a constant. Let $\mathcal{P}$ be $\{ab\}^+ \cup \{bird, ostrich\}^{\pm}$. Then, $\neg ostrich(tweety) \vee bird(tweety)$ belongs to $\mathcal{P}$, while $\neg ab(tweety)$ does not.

Definition 2.3 Let T be a set of formulas, F a formula, and $\mathcal{P}$ a production field.

1. The *characteristic clauses* of T are:
$$Carc(T) = \mu\,[Th_{\mathcal{P}}(T)]^1,$$
where for a set of clauses Σ, by $\mu\,[\Sigma]$ we mean the set of clauses of Σ not subsumed by any other clause of Σ.
2. The *new characteristic clauses of F with respect to T* are:
$$Newcarc(T, F) = Carc(T \cup \{F\}) - Carc(T),$$
that is, those characteristic clauses of $T \cup \{F\}$ that are not characteristic clauses of T.

Example 2.2 (continued) Let T be

$bird(tweety)$,
$\neg bird(tweety) \vee ab(tweety) \vee flies(tweety)$,
$\neg ostrich(tweety) \vee \neg flies(tweety)$.

In this well-known example, $P = \{ab\}$, $Z = \{flies\}$, and $Q = \{bird, ostrich\}$.

Let us fix as above $\mathcal{P}$ to be $P^+ \cup Q^{\pm}$, that is, positive occurrences of ***ab***, or any occurrence of ***bird*** and ***ostrich***. Then,

$$Carc(T) = \{\, bird(tweety),\ ab(tweety) \vee \neg ostrich(tweety)\,\},$$
$Newcarc(T, bird(tweety)) = \phi$
because $bird(tweety) \in Carc(T)$,
$Newcarc(T, flies(tweety)) = \{\neg ostrich(tweety)\}$
as $\neg ostrich(tweety) \notin Carc(T)$ belongs to $Carc(T \cup \{flies(tweety)\})$.

There is a strong connection between the concept of the new characteristic clauses and a logical account of abductive or hypothetical reasoning defined by such as [Poole *et al.*, 1987; Poole, 1989].

Definition 2.4 Let T be a set of formulas, D a set of ground literals (called the ***hypotheses***), and F a closed formula. A conjunction E of elements of D is an *explanation of F from (T, D)* if: (i) $T \cup \{E\}$ is satisfiable, and (ii) $T \cup \{E\} \models F$.

An explanation E of F from (T, D) is *minimal* if no proper sub-conjunct E' of E satisfies $T \cup \{E'\} \models F$.

[1] $Carc(T)$ depends on the production field $\mathcal{P}$, and thus a correct notation would be $Carc(T, \mathcal{P})$. As there will be no confusion about $\mathcal{P}$, we simply write $Carc(T)$.

An *extension of* (T, D) is the set of logical consequences of $T \cup \{E\}$ where E is a maximal conjunct of elements of D such that $T \cup \{E\}$ is satisfiable.

We will denote by $\neg \cdot \Sigma$ the set formed by taking the negation of each element in Σ.

Theorem 2.5 Let T, D and F be the same as Definition 2.4. The set of all minimal explanations of F from (T, D) is $\neg \cdot Newcarc(T, \neg F)$, where $\mathcal{P} = \neg \cdot D$.

Corollary 2.6 Let T, D and F be the same as Definition 2.4. There is no extension of (T, D) in which F holds iff there is no explanation of F from (T, D) iff $Newcarc(T, \neg F) = \phi$.

2.1 Przymusinski's Results

Przymusinski's [1989] algorithm is based on the following two theorems developed by Gelfond *et al.* [1989].

Theorem 2.7 [Przymusinski, 1989, Theorem 2.5] If a formula F does not contain literals from Z, then $CIRC(T; P; Z) \models F$ iff there is no clause E such that (i) E does not contain literals in $Z^{\pm} \cup P^-$, and (ii) $T \models \neg F \vee E$ but $T \not\models E$.

Now let us rewrite this theorem using the notation introduced above. Condition (i) means that E belongs to the production field $\mathcal{P} = P^+ \cup Q^{\pm}$. $T \models \neg F \vee E$ can be written as $T \cup \{F\} \models E$. So we are looking for a clause E belonging to the production field, implied by $T \cup \{F\}$ but not by T alone. This means that $E \in Th_{\mathcal{P}}(T \cup \{F\}) - Th_{\mathcal{P}}(T)$. The theorem requires that such E does not exist. Now, for a set of clauses Σ, $\Sigma = \phi$ iff $\mu[\Sigma] = \phi$. Therefore, by Lemma A.1, it is enough to check whether $Newcarc(T, F)$ is empty or not. That is,

Theorem 2.7 (new version) Let F be a formula not containing literals from Z. Let $\mathcal{P}$ be $P^+ \cup Q^{\pm}$. Then
$$CIRC(T; P; Z) \models F \quad \text{iff} \quad Newcarc(T, F) = \phi.$$

This formulation helps to understand the intuition underlying the above theorem. We want to know if a query F not involving literals from Z is true or not in the (P, Z)-minimal models of a theory T. Now, every (P, Z)-minimal model of T is defined on interpretations of T by considering differences of extensions of P and equality of extensions of Q, but by ignoring differences of extensions of Z [Lifschitz, 1985]. Therefore, the characteristic clauses of T are representative of those minimal models, in the sense that if adding F to T produces a change (new one) in $Carc(T)$ then the addition of F has produced a change in the minimal models of T as well. The existence of a new characteristic clause of F means that F has altered the minimal models: thus if $Newcarc(T, F)$ is empty, the addition of F has no effect on the minimal models and the circumscriptive theory entails it.

For formulas containing predicates from Z, the following holds:

Theorem 2.8 [Przymusinski, 1989, Theorem 2.6] Let F be any formula. $CIRC(T; P; Z) \models F$ iff either $T \models F$ or there is a formula G such that (i) G does not contain literals in $Z^{\pm} \cup P^-$, (ii) $T \models F \vee G$, and (iii) $CIRC(T; P; Z) \models \neg G$.

Now, $T \models F$ means $T \cup \{\neg F\}$ is unsatisfiable; note that in this case, $Newcarc(T, \neg F)$ would contain only $\Box$ (the empty clause). Condition (i) again means that G belongs to $\mathcal{P} = P^+ \cup Q^\pm$; condition (ii) can be written as $T \cup \{\neg F\} \models G$; and condition (iii) is equivalent to $Newcarc(T, \neg G) = \phi$ by Theorem 2.7. In this case, the condition $T \not\models G$ is missing in Theorem 2.8; if $T \models G$, however, then $Newcarc(T, \neg G) = \{\Box\} \neq \phi$ holds for satisfiable T. Therefore, condition (ii) together with (iii) further implies that G is of the form of a conjunction of clauses of $Newcarc(T, \neg F)$ [2]. And in condition (iii), if G is $\Box$, then $\neg G$ is the formula *true* and adding it to T produces no new theorem: $Newcarc(T, true) = \phi$. We can now write:

Theorem 2.8 (new version) Let F be any formula, and $\mathcal{P} = P^+ \cup Q^\pm$. Then $CIRC(T; P; Z) \models F$ iff there is a conjunct G of clauses in $Newcarc(T, \neg F)$ such that $Newcarc(T, \neg G) = \phi$.

While this formulation seems simpler than the original one, it still does not provide much insight. We will see it more clearly in Section 2.2, relating it with hypothetical reasoning. Let us review one of Przymusinski's examples with these new concepts.

Example 2.2 (continued) Przymusinski [1989, Example 3.10] shows that $CIRC(T; P; Z)$ does not imply $F_1 = flies(tweety)$ but implies $F_2 = ostrich(tweety) \vee flies(tweety)$. Let us verify these facts.

Adding $\neg F_1 = \neg flies(tweety)$ to T gives

$$Newcarc(T, \neg F_1) = \{ab(tweety)\}.$$

Since adding $\neg ab(tweety)$ to T gives a new characteristic clause, $\neg ostrich(tweety)$, $CIRC(T; P; Z) \not\models F_1$ holds.

Now we add

$$\neg F_2 = \neg ostrich(tweety) \wedge \neg flies(tweety)$$

to T, which gives

$$Newcarc(T, \neg F_2) = \{\neg ostrich(tweety),\ ab(tweety)\}.$$

The negation of the conjunction of these two clauses is

$$ostrich(tweety) \vee \neg ab(tweety)\,.$$

Adding this formula to T produces no new characteristic clauses, as the only new theorems are

$$\{ostrich(tweety) \vee \neg ab(tweety),\ flies(tweety)\},$$

and neither belongs to $\mathcal{P}$. Thus, as expected, F_2 is in the circumscribed theory.

2.2 Ginsberg's Results

Ginsberg [1989] presents an another algorithm for computing circumscription. The algorithm however works only in the case where Q, the set of fixed predicates, is empty. We will transform Ginsberg's definitions and results to ours.

[2]In practice, the minimality condition involved by the μ operation is not crucial. See Section 3.1.

Definition 2.9 [Ginsberg, 1989, Definition 3.1] Let D and T be two sets of formulas. G is *dnf wrt* D if it is written as a disjunction of conjunctions of elements of D. And F is *confirmed by* G (wrt T and D) if the following conditions hold:

1. $T \cup \{G\}$ is satisfiable,
2. $T \cup \{G\} \models F$, and
3. G is dnf wrt D.

Comparing Definition 2.9 with Definition 2.4, we see that F is confirmed by G wrt D if G is a disjunction of explanations of F from (T, D). Now, $\neg G$ is a conjunction of clauses belonging to the production field $\neg \cdot D$ by Theorem 2.5. Or, in other words,

Definition 2.9 (new version) Let $\mathcal{P} = \neg \cdot D$. F is confirmed by G if $\neg G$ is a conjunction of clauses in $Newcarc(T, \neg F)$. Moreover, F is *unconfirmed*, if no G confirms F: $Newcarc(T, \neg F) = \phi$.

Next is the main result:

Proposition 2.10 [Ginsberg, 1989, Proposition 3.2] Let D be P^-. $CIRC(T; P; Z) \models F$ iff there is some G confirming F so that $\neg G$ is unconfirmed.

We can rewrite it as:

Proposition 2.10 (new version) Let $\mathcal{P} = \neg \cdot D$. $CIRC(T; P; Z) \models F$ iff there is a conjunct G of clauses in $Newcarc(T, \neg F)$ such that $Newcarc(T, \neg G) = \phi$.

Ginsberg briefly mentions connections with Przymusinski's work and the possibility of relaxing the assumption of all non-minimized predicates being variable. Our above results show that:

1. This last proposition is exactly Theorem 2.8.
2. All results can thus be extended to the case $Q \neq \phi$ (that is, not varying all predicates) just by setting $D = P^- \cup Q^\pm$, that is, $\mathcal{P} = \neg \cdot D = P^+ \cup Q^\pm$.

The intuition behind Theorem 2.8 and Proposition 2.10 is the following. From the viewpoint of abductive reasoning, those theorems say that $CIRC(T, P, Z) \models F$ iff there is a disjunct G of explanations from (T, D) such that there exists no explanation of $\neg G$ from (T, D) [3], and Poole [1989] introduces the similar condition for a formula to hold in all extensions of (T, D)[4]. For answering queries in circumscription, the hypotheses D must be carefully chosen in the direction of (P, Z)-minimization: for minimized predicates P, P^- should be hypothesized, and for fixed predicates Q, $Q^\pm$ should be taken into account. Now the existence of an explanation of F from (T, D) guarantees that F holds in at least one extension of (T, D) by Corollary 2.6. Clearly, if some disjunct G of explanations of F holds in all extensions, then F also holds in all extensions. Since this G is constructed over D and thus does not contain literals from Z, we see that

[3]Lin & Goebel [1989] independently derive the equivalent theorem from the result by [Gelfond *et al.*, 1989] within the Theorist framework [Poole *et al.*, 1987].

[4]Etherington [1987] has shown the equivalence of membership in all extensions and circumscriptive entailment for propositional theories without fixed predicates.

G holds in all extensions of (T, D) iff $Newcarc(T, G) = \phi$ wrt $\mathcal{P} = \neg \cdot D$ (by Theorem 2.7) iff there is no explanation of $\neg G$ from (T, D) (by Corollary 2.6).

3 Comparing the Algorithms

In the last section we showed that both Przymusinski's and Ginsberg's algorithms were based on the same theoretical results. This section concerns the computational efficiency of the algorithms.

Przymusinski [1989] defines *MILO-resolution*, a variant of ordered linear (OL) resolution [Chang and Lee, 1973]. Given a clause C, MILO-resolution is used to deduce a set of minimal clauses belonging to $Th_{\mathcal{P}}(T \cup \{C\})$, called the *derivative of* $T + C$, with top clause C and the background theory T. The algorithm needs to check the non-deducibility of each clause in the derivative from T, in order to determine the new characteristic clauses. On the other hand, Ginsberg's [1989] circumscriptive theorem prover uses a "backward-chaining ATMS" [Reiter and de Kleer, 1987] to compute minimal explanations of formulas. This backward chaining procedure also uses a classical theorem prover.

While the structure of the proofs are similar, each algorithm has a different concern and extends a resolution procedure in a different way. Remember that we should produce clauses (i) in the production field, and (ii) the "new" and "minimal" of these, that is, neither implied by the original theory nor by another produced clause. MILO-resolution provides the ability to restrict the resolution to some literals by which the algorithm directly focuses on producing the clauses relevant to answer the query, that is, those in the production field $P^+ \cup Q^\pm$. Przymusinski's concern is thus efficiency regarding the first of the above two points. Ginsberg uses a classical theorem prover; this means that no information concerning the production field is used during the proof. His algorithm has however another concern, that of the minimality of the produced formulas. For this, he uses a structure called a "bilattice" based on his previous work on multivalued logic [Ginsberg, 1988]. The role of this bilattice is to record inferences, in order to avoid making them more than once. He is thus concerned with the second of the above.

The next two subsections expand on these ideas. The discussion is based on each resolution procedure to compute $Newcarc(T, C)$, given a background theory T, a clause C and a production field $\mathcal{P}$.

3.1 What Needs to be Computed

From the results presented above, it appears that to answer a query, an algorithm should first compute the minimal explanations of a formula F from $(T, P^- \cup Q^\pm)$, or equivalently their negations, $Newcarc(T, \neg F)$, with the production field set to $P^+ \cup Q^\pm$. Ginsberg's theorem prover works exactly along this line of computation [5].

However, there is a set smaller than $Newcarc(T, F)$ that can be used to answer such a query. Let us divide the produced clauses $\mathcal{S}$ by using deductions with top clause C, the background theory T and the production field $\mathcal{P}$, possibly containing subsumed clauses (note that $Newcarc(T, C) \subseteq \mathcal{S}$; see Theorem A.3) into two sets, say $\mathcal{S}_1$ and $\mathcal{S}_2$, such that

$$\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \text{ and } T \cup \mathcal{S}_1 \models \mathcal{S}_2 .$$

Adding $\mathcal{S}_2$ to $\mathcal{S}_1$ does not change the models of the produced clauses, so only $\mathcal{S}_1$ needs to be computed model-theoretically. We call a set $\mathcal{S}_1$ verifying this condition a *precursor of* $\mathcal{S}$. Note that a clause in a precursor may not belong to $Newcarc(T, \mathcal{P})$, that is, the clause is not always minimal in the sense of set-inclusion, but it is the weakest in the sense that for any clause $A_2 \in \mathcal{S}_2$ there exists a clause $A_1 \in \mathcal{S}_1$ such that $T \cup \{\neg A_2\} \models \neg A_1$ holds (recall that for $A \in \mathcal{S}$, $\neg A$ is an explanation of $\neg C$ from $(T, \neg \cdot \mathcal{P})$ if it is consistent with T [6]).

MILO-resolution actually computes such a precursor, as the derivative of $T + C$, because it restricts the resolution to literals belonging to $Z^\pm \cup P^-$. In other words, when the first literal of the center clause belongs to $\mathcal{P} = P^+ \cup Q^\pm$, it is *skipped*. If it were resolved upon with a clause from the theory, the resulting leave obtained by chaining the inference would be implied by the one obtained with the skipping operation (see Theorem 4.2). This is best understood with an example.

Example 3.1 The theory is

$$T = \{ p_1 \vee \neg p_2, \;\; p_2 \vee \neg p_3, \;\; p_3 \vee z_1 \} .$$

The production field is $\mathcal{P} = P^+ = \{p_1, p_2, p_3\}^+$. The query is z_1.

By adding $\neg z_1$ to T, MILO-resolution generates only p_3, the only new theorem that belongs to $\mathcal{P}$. Since this literal belongs to $\mathcal{P}$, the procedure skips it and stops. It then adds $\neg p_3$ to T which generates no characteristic clause, showing that $CIRC(T; P; Z) \models z_1$.

If the procedure would examine the remaining choice, resolving p_3 with the clause $p_2 \vee \neg p_3$, it would produce p_2, and a further step would produce p_1. This is exactly what Ginsberg's prover does, as it uses no information from $\mathcal{P}$ to stop the execution when p_3 is produced. The set of assumptions is $D = \neg \cdot \mathcal{P} = P^-$. The confirmation of z_1 produced by an ATMS is dnf:

$$\neg p_3 \vee \neg p_2 \vee \neg p_1 .$$

The negation of the confirmation is

$$p_3 \wedge p_2 \wedge p_1 ,$$

which is unconfirmed. Using Ginsberg's terminology, two additional contexts have been produced, $\{\neg p_1\}$ and $\{\neg p_2\}$, in which z holds (the three are produced because none of them is a subset of another). MILO-resolution did not need to generate them. As explained above, the reason is that $\{p_3\}$ is the precursor of the others, as:

$$T \cup \{p_3\} \models p_1 \wedge p_2 .$$

There is another big difference between Przymusinski's and Ginsberg's provers concerning checking the

[5] This is in essence what [Lin and Goebel, 1989] does too.

[6] An explanation E_1 is called *less-presumptive than* E_2 [Poole, 1989] if $T \cup \{E_2\} \models E_1$. Therefore, an explanation in $\neg \cdot \mathcal{S}_1$ is a least-presumptive explanation of $\neg C$ from $(T, \neg \cdot \mathcal{P})$.

consistency of hypotheses. Recall that to apply Theorem 2.8 or Proposition 2.10, we need two steps; firstly computing a set of clauses belonging to $Newcarc(T, \neg F)$ for the query F, then checking whether a conjunct G of those clauses satisfies $Newcarc(T, \neg G) = \phi$. Ginsberg's prover first computes the minimal explanations $\mathcal{E}$ of F from $(T, \neg \cdot \mathcal{P})$, then computes the minimal explanations of $\neg \bigvee_{E \in \mathcal{E}} E$ from $(T, \neg \cdot \mathcal{P})$; on the contrary, Przymusinski's prover first computes the derivative $\mathcal{D}_1$ of $T + F$, without checking the non-deducibility of each clause in $\mathcal{D}_1$ from T, then computes the derivative $\mathcal{D}_2$ of $T + \neg \bigvee_{A \in \mathcal{D}_1} A$, checking whether $T \not\models B$ for each clause B in $\mathcal{D}_2$ one by one. Clearly, in the second step, we need not compute all the minimal explanations for the negation of the disjunct; if it has at least one explanation then we can stop the computation immediately. For the first step, Przymusinski's prover may include some clauses belonging to $Th_{\mathcal{P}}(T)$ in $\mathcal{D}_1$, which are excluded from the produced clauses by Ginsberg's prover [7]. However, since it takes much computation for this consistency checking (non-decidable for the first-order case), it seems rather efficient even if these extra clauses are taken into account in the second step (indeed, the efficiency may depend on the knowledge base).

3.2 How it is Computed

Now suppose both algorithms have to compute the same set, that is, MILO-resolution computes all the new characteristic clauses by avoiding the skipping of literals, or Ginsberg's theorem prover is restricted to computing a precursor. In that case, another advantage of using the information on the production field $\mathcal{P}$ during the deduction is that fewer clauses are generated.

For a very simple example, suppose that the center clause is $z \vee q$, where $z \notin \mathcal{P}$, while $q \in \mathcal{P}$. If z cannot be resolved upon against clauses of the theory in such a way that the result of the deduction produces a clause belonging to $\mathcal{P}$, MILO-resolution will never try to resolve on the next literal q. Conventional theorem provers will give no priority to z over q and thus will try all the resolutions on q as well, making unnecessary computation. This example, although trivial, is representative of what will happen in many realistic situations.

We said above that a central concern of [Ginsberg, 1989] was to avoid computing the same clauses more than once. The role of the bilattice is to record information and use it to avoid redundant derivations by making subsumption tests. Avoiding the exploration of unnecessary portions of the search space, and in particular the non-production of subsumed clauses has been a central concern of automated theorem proving and is one of the motivations behind all the refinements of resolution [Loveland, 1978]. Many of these use the information of literals that have been resolved upon to avoid producing many of redundant clauses. For example, OL-resolution on which MILO-resolution uses *framed literals* to record the history of the deduction. Regarding other problems related to irredundancy and control, a thorough analysis can be found in the chapter on subsumption in [Loveland, 1978]. There is not enough information in [Ginsberg, 1989; Ginsberg, 1988] to determine whether the bilattice represents a better alternative.

[7]If there is a clause $A \in \mathcal{D}_1$ such that $T \models A$, then in the second step, the negation of the disjuncts contains $\neg A$. Since its valuation is false, this does not affect the result of the query answering.

4 Improving Efficiency

We show here how MILO-resolution's search space can be reduced. MILO-resolution is based on Chang and Lee's [1973] version of OL-resolution; this procedure is augmented with the ability to skip literals when they belong to the production field.

Now, actually there exist superior versions of linear resolution that can be augmented with skipping operations. Most notably, Model Elimination [Loveland, 1978] and SL-resolution [Kowalski and Kuhner, 1971]. Basically, the Model Elimination procedure introduced the restriction that without loss of completeness, it can avoid resolving the center clause with clauses from the theory that have literals equal to framed literals at the right of the literal resolved upon, as this would produce only clauses subsumed by some previous center clause.

Example 4.1 Suppose a clause,

$$F = a \vee d\,,$$

resolves with a clause,

$$\neg a \vee b\,,$$

in T, giving

$$b \vee [a] \vee d\,.$$

Now suppose there is a clause,

$$\neg b \vee a\,,$$

in T. The above restriction tells us that this clause may safely be skipped in the deduction, as it contains a, a literal appearing framed in the center clause. In effect, it would give the clause,

$$a \vee [b] \vee [a] \vee d\,,$$

which is subsumed by a previous center clause, F.

Clearly, this has two advantages: it restricts the search space, and avoids many of subsumption tests in step (iii) of MILO-resolution [Przymusinski, 1989, Definition 3.1].

Additional improvements in efficiency were introduced by [Shostak, 1976] and [Bibel, 1982]. Shostak [1976] shows that we can record still more information on center clauses. When the first literal of the center clause is framed, previous versions of linear resolution delete it from the clause. Shostak's procedure complements it and keeps it in a different position called the *C-point* in the clause, where it can still be used later in the reduction to reduce the search space and do ancestor resolution as with ordinary framed literals.

Example 4.1 (continued) If now the above clause,

$$b \vee [a] \vee d\,,$$

is resolved with,

$$\neg b\,,$$

the result will be

$$d \vee (\neg b) \vee (\neg a),$$

where, the notation $(\neg l)$ is used for the truncated framed literal $[l]$ moved to the C-point; thus the information that $\neg a$ and $\neg b$ are proved is kept.

4.1 Summary

We thus propose the following procedure schema. Given a set of clauses T, a clause C, and a production field $\mathcal{P}$, a deduction of a clause K from $T + C$ (the background theory T with top clause C) and $\mathcal{P}$ consists of a sequence of structured clauses, $C_0, C_1, \ldots, C_n$, such that:

1. $C_0 = \langle \Box,\ C \rangle$,
2. $C_n = \langle K,\ \Box \rangle$, and
3. $C_{i+1} = \langle K_{i+1},\ R_{i+1} \rangle$ is obtained from $C_i = \langle K_i,\ R_i \rangle$ by applying either of the following operations. We assume that R_i is ordered and l is the first literal of R_i.
   (a) **(Skip)** If $l \in \mathcal{P}$, then $K_{i+1} = K_i \vee l$ and R_{i+1} is obtained by removing l from R_i.
   (b) Otherwise, $K_{i+1} = K_i$ and R_{i+1} is obtained by a linear resolution procedure where the center clause is R_i and the background theory is T.

By using this procedure we can find a precursor without computing all the $Newcarc(T, C)$:

Theorem 4.2 If a clause L belongs to $Newcarc(T, C)$, then there is a deduction of a clause $M \in Th_{\mathcal{P}}(T \cup \{C\})$ from $T + C$ and $\mathcal{P}$ such that $T \cup \{M\} \models L$.

The query answering procedure for circumscriptive theories [Przymusinski, 1989, Algorithm 4.1] that calls MILO-resolution remains identical; the definition of the derivative of $T + C$ is just changed to be the output of the above procedure instead of the output of MILO-resolution.

4.2 Example

We might use Shostak's GC procedure as a linear resolution procedure for step 3(b) above, and get the following:

Example 4.3 (modified version of [Przymusinski, 1989, Example 3.5]) We apply the procedure to formulas with variables in order to show that it is not limited to the ground case. Let T contain the following formulas, with $P = \{learns, senior\}$, and $Z = \phi$.

$$\forall X\ senior(X) \supset learns(X, latin) \vee learns(X, greek),$$
$$\forall X\ \ senior(X) \supset learns(X, french),$$
$$senior(ann),$$
$$\forall X\ \ learns(X, greek) \supset senior(X).$$

In clausal form and with the obvious abbreviations, the theory is

$$l(X, lt) \vee l(X, gr) \vee \neg s(X), \qquad (1)$$
$$\neg s(X) \vee l(X, fr), \qquad (2)$$
$$s(a), \qquad (3)$$
$$\neg l(X, gr) \vee s(X). \qquad (4)$$

The production field $\mathcal{P}$ is then $P^+ = \{l, s\}^+$. Consider the query $\neg l(a, gr) \vee \neg l(a, fr)$. The following is a deduction obtained by our procedure.

$C_0 = \langle \Box,\ \neg l(a, gr) \vee \neg l(a, fr) \rangle$ —Given.
$C_1 = \langle \Box,\ l(a, lt) \vee \neg s(a) \vee [\neg l(a, gr)] \vee \neg l(a, fr) \rangle$ —Resolution with 1 (*).
$C_2 = \langle l(a, lt),\ \neg s(a) \vee [\neg l(a, gr)] \vee \neg l(a, fr) \rangle$ —Skip the literal from $\mathcal{P}$ (**).
$C_3 = \langle l(a, lt),\ [\neg s(a)] \vee [\neg l(a, gr)] \vee \neg l(a, fr) \rangle$ —Resolution with 3.
$C_4 = \langle l(a, lt),\ \neg l(a, fr) \vee (s(a)) \vee (l(a, gr)) \rangle$ —Recording of solved literals.
$C_5 = \langle l(a, lt),\ \neg s(a) \vee [\neg l(a, fr)] \vee (s(a)) \vee (l(a, gr)) \rangle$ —Resolution with 2.
$C_6 = \langle l(a, lt),\ [\neg l(a, fr)] \vee (s(a)) \vee (l(a, gr)) \rangle$ —Truncation using the solved literal (***).
$C_7 = \langle l(a, lt),\ \Box \rangle$ —Reduction.

Now according to Theorem 2.7, we can answer "no" to the above query, that is,

$$CIRC(T; P; Z) \not\models \neg l(a, gr) \vee \neg l(a, fr),$$

because $l(a, lt)$ is not implied by T.

Let us look at some advantages of this deduction over MILO-resolution and Ginsberg's theorem prover.

- At point (*), Ginsberg's prover, which lacks information on the production field, will resolve on $l(a, lt)$, instead of skipping it, thus exploring branches that are pruned by the skipping operation.
- At point (**), MILO-resolution will behave as in the above deduction, but keeps an additional choice that results from resolving $\neg s(a)$ with clause 4. Our procedure avoids this because clause 4 contains the literal $\neg l(a, gr)$ that appears framed in the center clause, thus indicating the remaining choice is unnecessary.
- At point (***), MILO-resolution would have lost information about $s(a)$, and thus makes a resolution against all clauses containing $\neg s(a)$. Reduction with solved literal avoids this exploration.

5 Concluding Remarks

We have compared two algorithms to compute circumscription, relating them to abductive reasoning, and showing that they are based on the same theoretical results. We have also analyzed their computational properties, showing their different concerns: [Przymusinski, 1989] defines the set of formulas that needs to be computed and uses skipping operations to compute them directly; [Ginsberg, 1989] concerns avoiding redundancy by recording information during a deduction.

The skipping operation can be applied to other, more efficient versions of linear resolution, and further improvements on these methods can be incorporated into the procedure. Other techniques of theorem proving can be used to improve efficiency still more. For example, we can "compile" the theory, producing either its prime implicates [Reiter and de Kleer, 1987] or the sub-clauses implied by it. In both cases, the resultant theory has the same models, and thus the same (P, Z)-minimal models as the former. Deduction from this compiled theory will give the same results as from the former.

The improvements in the present paper are based on direct refinements of linear resolution procedures, and actually applicable to efficient computation of $Newcarc(T, \neg F)$ for a query F and $Newcarc(T, \neg G)$ for some G in Theorem 2.8. However, both Przymusinski's [1989] and Ginsberg's [1989] algorithms are naive implementations of Theorem 2.8, which states the need for the existence of a certain conjunct G of $Newcarc(T, \neg F)$, ignoring that many of clauses in $Newcarc(T, \neg F)$ may exist. Therefore they turn out to suffer from the following two problems in their computations:

1. Both "algorithms" do not work for the case in which there are potentially an *infinite number of* clauses in $Newcarc(T, \neg F)$. Even if the number of $Newcarc(T, \neg F)$ is finite, not all of them are the relevant parts needed to determine that F is in the circumscribed theory [8].
2. Neither algorithm can handle the *answer extraction for open queries*. That is, when a query contains variables, the algorithms cannot return the substitution values of the variables for which the query holds. This is a much broader problem than "Yes/No" type questions.

A procedure attempting to solve the first problem is proposed by Poole [1989], which is the dialectical implementation of membership in all extensions. In [Helft *et al.*, 1989], which complements this paper, a solution for both the first and the second problems was proposed, which finds a minimal, rather than maximal conjunct G of $Newcarc(T, \neg F)$. While we will not discuss it further in this paper, we should note that a certain subset of $Newcarc(T, \neg F)$ must be computed anyway. Therefore, the improvements proposed in this paper can still be applied to any proof procedure attempting to solve these problems.

Acknowledgment

We would like to thank David Poole, Koichi Furukawa and Randy Goebel for their helpful discussions on this topic and their useful comments on an earlier version of this paper. We are also grateful to Mark Stickel for his comments about improving efficiency of proof procedures.

[8] While Przymusinski's prover does not compute all of $Newcarc(T, \neg F)$ but computes a precursor of them, in some cases the precursor may have potentially infinite clauses especially for the first-order case.

References

[Bibel, 1982] Wolfgang Bibel. A Comparative Study of Several Proof Procedures. *Artificial Intelligence* **18** (1982) 269–293.

[Bossu and Siegel, 1985] Genevieve Bossu and Pierre Siegel. Saturation, Nonmonotonic Reasoning, and the Closed-World Assumption. *Artificial Intelligence* **25** (1985) 23–67.

[Chang and Lee, 1973] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, New York, 1973.

[Etherington, 1987] David W. Etherington. Relating Default Logic and Circumscription. *Proc. IJCAI-87,* Milan (1987) 489–494.

[Gelfond *et al.*, 1989] Michael Gelfond, Halina Przymusinska, and Teodor Przymusinski. On the Relationship between Circumscription and Negation as Failure, *Artificial Intelligence* **38** (1989) 75–94.

[Ginsberg, 1988] Matthew L. Ginsberg. Multivalued Logics: A Uniform Approach to Reasoning in Artificial Intelligence. *Computational Intelligence* **4** (1988) 265–316.

[Ginsberg, 1989] Matthew L. Ginsberg. A Circumscriptive Theorem Prover. *Artificial Intelligence* **39** (1989) 209–230.

[Helft *et al.*, 1989] Nicolas Helft, Katsumi Inoue, and David Poole. Extracting Answers in Circumscription. ICOT Technical Memorandum TM-855, ICOT, Tokyo, 1989.

[Kowalski and Kuhner, 1971] Robert A. Kowalski and D. G. Kuhner. Linear Resolution with Selection Function. *Artificial Intelligence* **2** (1971) 227–260.

[Lifschitz, 1985] Vladimir Lifschitz. Computing Circumscription. *Proc. IJCAI-85,* Los Angeles (1985) 121–127.

[Lin and Goebel, 1989] Dekang Lin and Randy Goebel. Computing Circumscription of Ground Theories with Theorist. Technical Report TR 89-26, Department of Computer Science, The University of Alberta, Edmonton, 1989.

[Loveland, 1978] Donald W. Loveland. *Automated Theorem Proving: A Logical Basis.* North-Holland, Amsterdam, 1978.

[McCarthy, 1980] John McCarthy. Circumscription—A Form of Non-Monotonic Reasoning. *Artificial Intelligence* **13** (1980) 27–39.

[Minicozzi and Reiter, 1972] Eliana Minicozzi and Raymond Reiter. A Note on Linear Resolution Strategies in Consequence-Finding. *Artificial Intelligence* **3** (1972) 175–180.

[Poole, 1989] David Poole. Explanation and Prediction: An Architecture for Default and Abductive Reasoning. *Computational Intelligence* **5** (1989) 97–110.

[Poole *et al.*, 1987] David Poole, Randy Goebel and Romas Aleliunas. Theorist: A Logical Reasoning System for Defaults and Diagnosis. In: Nick Cercone and Gordon McCalla, editors, *The Knowledge Frontier: Essays in the Representation of Knowledge.* Springer-Verlag, New York (1987) 331–352.

[Przymusinski, 1989] Teodor C. Przymusinski. An Algorithm to Compute Circumscription. *Artificial Intelligence* **38** (1989) 49–73.

[Reiter and de Kleer, 1987] Raymond Reiter and Johan de Kleer. Foundations of Assumption-Based Truth Maintenance Systems: Preliminary Report. *Proc. AAAI-87,* Seattle (1987) 183–187.

[Shostak, 1976] Robert E. Shostak. Refutation Graphs. *Artificial Intelligence* **7** (1976) 51–64.

[Siegel, 1987] Pierre Siegel. Représentation et Utilisation de la Connaissance en Calcul Propositionnel. PhD thesis, University of Aix-Marseille II, Marseille, 1987.

A Appendix: Proofs of Theorems

The next lemma is used to prove Theorem 2.5.

Lemma A.1 Let T be a set of clauses, F a formula.

$$Newcarc(T, F) = \mu\,[Th_{\mathcal{P}}(T \cup \{F\}) - Th_{\mathcal{P}}(T)]\,.$$

Proof: Let $A = Th_{\mathcal{P}}(T \cup \{F\})$ and $B = Th_{\mathcal{P}}(T)$. Notice that $B \subseteq A$. We will prove that $\mu[A - B] = \mu[A] - \mu[B]$.

Let $c \in \mu[A - B]$. Then obviously $c \in A - B$ and thus $c \in A$. Now assume that $c \notin \mu[A]$. Then $\exists d \in A$ such that $d \subset c$. By the minimality of $c \in A - B$, $d \in B$. Since $d \subset c$, $c \in B$, contradiction. Therefore $c \in \mu[A]$. Clearly, by $c \notin B$, $c \notin \mu[B]$. Hence, $c \in \mu[A] - \mu[B]$.

Conversely, assume that $c \in \mu[A] - \mu[B]$. Firstly we must prove that $c \in A - B$. Suppose to the contrary that $c \in B$. Since $c \notin \mu[B]$, $\exists d \in B$ such that $d \subset c$. However, as $B \subseteq A$, $d \in A$, contradicting the minimality of $c \in A$. Therefore, $c \in A - B$. Now assume that c is not minimal in $A - B$. Then, $\exists e \in A - B$ such that $e \subset c$, again contradicting the minimality of $c \in A$. Hence, $c \in \mu[A - B]$. □

Theorem 2.5 Let T be a set of clauses, D a set of ground literal, F a formula. The set of all minimal explanations of F from (T, D) is $\neg \cdot Newcarc(T, \neg F)$, where $\mathcal{P} = \neg \cdot D$.

Proof: Now, suppose that E is an explanation of F from (T, D). By Definition 2.4, it is observed that (i) the fact that $T \cup \{E\}$ is satisfiable means $T \not\models \neg E$, (ii) $T \cup \{E\} \models F$ can be written as $T \cup \{\neg F\} \models \neg E$, and $\neg E$ is a clause all of whose literals belong to $\neg \cdot D$. Thus $\neg E \in Th_{\mathcal{P}}(T \cup \{\neg F\}) - Th_{\mathcal{P}}(T)$. By Lemma A.1, E is a minimal explanation of F from (T, D) iff $\neg E \in Newcarc(T, \neg F)$. □

We need the following preliminaries for the proof of Theorem 4.2. In the subsequent discussion, we will denote a clause as a set of literals. Firstly, a complete abductive procedure is defined by modifying the procedure described in Section 4.1 as follows:

Definition A.2 Given a set of clauses T, a clause C, and a production field $\mathcal{P}$, an *LS (Skipping Linear) deduction of a clause K from $T + C$ and $\mathcal{P}$* consists of a sequence of structured clauses, $C_0 = \langle \Box,\ C \rangle, \ldots, C_i, C_{i+1}, \ldots, C_n = \langle K,\ \Box \rangle$, such that $C_{i+1} = \langle K_{i+1},\ R_{i+1} \rangle$ is obtained from $C_i = \langle K_i,\ R_i \rangle$ by applying either of the following operations (we assume that R_i is ordered and l is the first literal of R_i):

3(a′) **(Skip)** If $l \in \mathcal{P}$, then $K_{i+1} = K_i \cup \{l\}$ and $R_{i+1} = R_i - \{l\}$.

3(b′) $K_{i+1} = K_i$ and R_{i+1} is obtained by a linear resolution procedure where the center clause is R_i and the background theory is T.

An example of LS resolution is proposed by Siegel [1987], which incorporates the restriction rule used in Example 4.1. The only difference between an LS deduction and one in Section 4.1 is that while the rules 3(a) and 3(b) in the latter are exclusive, in the former 3(a′) and 3(b′) are not; for $l \in \mathcal{P}$ either rule can be applied. The next theorem shows that LS resolution is complete for finding new characteristic clauses.

Theorem A.3 If a clause L belongs to $Th_{\mathcal{P}}(T \cup \{C\}) - Th_{\mathcal{P}}(T)$, then there is an LS deduction of a clause $M \in Th_{\mathcal{P}}(T \cup \{C\})$ from $T + C$ and $\mathcal{P}$ such that M subsumes L.

Proof: The proof can be seen as an extension of the completeness result for consequence-finding in linear resolution by Minicozzi & Reiter [1972] augmented with the skipping operation. And the result also follows easily using the same method as in the completeness proof for the procedure described in [Siegel, 1987]. □

By using Theorem A.3, we can show that $Newcarc(T, C)$ is a subset of the set of clauses derived using LS deductions from $T + C$ and $\mathcal{P}$. Now, we will prove that, if a clause L is derived by using an LS deduction from $T + C$ and $\mathcal{P}$, then there is an LS deduction of a clause M from $T + C$ and $\mathcal{P}$ by using only the **Skip** rule for each first literal $l \in \mathcal{P}$ in every center clause, such that $T \cup \{M\} \models L$. This result completes the proof of Theorem 4.2.

Theorem 4.2 If a clause L belongs to $Newcarc(T, C)$, then there is a deduction of a clause $M \in Th_{\mathcal{P}}(T \cup \{C\})$ from $T + C$ and $\mathcal{P}$ such that $T \cup \{M\} \models L$.

Proof: Let $C_0, C_1, \ldots, C_n$ be an LS deduction of L from $T + C$ and $\mathcal{P}$. Let l_i be the first literal of R_i, where $C_i = \langle K_i,\ R_i \rangle$ and $0 \le i \le n - 1$. Firstly, if **Skip** is applied for every l_j $(0 \le j \le n - 1)$ such that $l_j \in \mathcal{P}$, then L is actually derived from $T + C$ and $\mathcal{P}$, and of course $T \cup \{L\} \models L$ holds.

Next, suppose that $\exists C_j$ in the LS deduction such that $l_j \in \mathcal{P}$ is resolved upon with a clause $B_j \in T$. In the following proof, to simplify the discussion, we assume that there are no identical, truncated, or reduced literals in R_{y+1} and denote R_{y+1} by removing the framed literals in it; if they exist, then we can modify the proof appropriately. Let x $(1 \le x \le n)$ be the number of such clauses, and C_y be such a clause where y $(0 \le y \le n - 1)$ is the largest number. In this case, $C_{y+1} = \langle K_{y+1},\ R_{y+1} \rangle$, where $K_{y+1} = K_y$ and $R_{y+1} = (B_y - \{\neg l_y\}) \cup (R_y - \{l_y\})$. Now, let U be a clause LS derived from $T + (B_y - \{\neg l_y\})$ and $\mathcal{P}$, V a clause LS derived from $T + (R_y - \{l_y\})$ and $\mathcal{P}$. Here, we can choose such U and V to satisfy $L = K_y \cup U \cup V$, because L is LS derived from $T + (K_{y+1} \cup R_{y+1})$ and $\mathcal{P}$.

Now assume that instead of resolving R_y with B_y, **Skip** is applied to K_y, deducing $K'_{y+1} = \langle K'_{y+1},\ R'_{y+1} \rangle$, where $K'_{y+1} = K_y \cup \{l_y\}$ and $R'_{y+1} = R_y - \{l_y\}$. Then, $K_y \cup \{l_y\} \cup V$ is LS derived from $T + (K'_{y+1} \cup R'_{y+1})$ and $\mathcal{P}$, and thus from $T + C$ and $\mathcal{P}$. Since $T \cup \{l_y\} \models B_y - \{\neg l_y\}$, $T \cup \{l_y\} \models U$ holds, and thus $T \cup \{(K_y \cup \{l_y\} \cup V)\} \models L$ holds.

Now let $M_0 = L$ and $M_1 = (K_y \cup \{l_y\} \cup V)$. In the similar way, we can find an LS deduction of M_2 from $T + C$ and $\mathcal{P}$ such that $T \cup \{M_2\} \models M_1$, by resetting y to the second largest number. By using the bottom-up manner, we can successively find clauses M_j $(1 \le j \le x)$ LS derived from $T + C$ and $\mathcal{P}$ such that $T \cup \{M_j\} \models M_{j-1}$. Therefore, $T \cup \{M_x\} \models M_{x-1}$, $T \cup \{M_{x-1}\} \models M_{x-2}$, $\ldots$, $T \cup \{M_1\} \models M_0$. Hence, $T \cup \{M_x\} \models M_0$, and we get the theorem. □

Improving Deduction in a Sequent Calculus

Catherine Belleannée

IRISA
Campus de Beaulieu 35042 Rennes Cedex
FRANCE

Abstract

This paper deals with a natural-like method of deduction for First-Order Logic: the *system G* of Gentzen. This system is, among natural-like methods, one of the best suited to mechanisation, due to the use of both a sequent formalism, providing control facilities for deduction, and a system of rules being "convergent". The paper describes two complementary improvements of the system G to make it even more natural as well as more efficient.
The first one deals with an extension of the set of inference rules. Actually, for some formulas, the use of new connectives (not contained in the standard system) has interesting consequences. Directly handling these new connectives through the use of macro-rules increases both the performances of the system (saving time and space) and the naturalness of the proofs.
The second improvement concerns the search procedure. It consists to detect dynamically whether useless steps are being performed, and, if so, to stop their development. As above, this improvement leads to gain in both efficiency due to the decrease in the number of developed nodes in the deduction tree, and naturalness of the resulting proofs, since useless steps have been removed.

1 Introduction

Out of all existing methods of deduction, ***resolution*** emerges as a proof procedure dealing with formulas in a standard form by means of a single inference rule ([Chang and Lee, 73] [Stickel, 88]). In contrast, ***natural-like methods*** accept formulas in their original form, and so, use more than one inference rule in order to take into account the different connectives. This paper deals with a system of the latter kind: the ***system G***, a sequent Calculus defined by Gentzen ([Gallier, 86, Bowen, 82]).
Actually the system G is, among natural-like methods, one of the best suited to mechanisation. First of all, the sequent method keeps track of all the formulas available at each deduction step, thus providing facilities to control the deduction. This contrasts with the method of the ***semantic tableaux*** for instance, which spreads formulas along the deduction tree ([Smullyan, 68]). Furthermore, the inference rules of the system G are "convergent": they make the number of connectives in the sequent to decrease (except for some rules concerning quantifiers), and then the search procedure is trivially decidable for propositional calculus. This differs from the natural deduction systems, where we must face the harder problem of controlling both the use of introducing and discharging assumptions. This paper describes two complementary improvements of the system G to make it even more natural as well as more efficient.
The first section briefly introduces the features of the original system G, the following section deals with an extension of the set of inference rules, and the last one concerns a modification of the search procedure.

2 System G

2.1 Formal system

2.1.1 Sequents

Definition : A *sequent* is a pair (Γ, Δ) of, possibly empty, sets of formulas.
Notation: the sequent (Γ, Δ) is usually noted $\Gamma \rightarrow \Delta$, and if $\Gamma = \{A1, ..., Am\}$ and $\Delta = \{B1, ..., Bn\}$ the simplified notation is $A1, .., Am \rightarrow B1, .., Bn$.

Semantics : Given an interpretation I, a sequent $A1, .., Am \rightarrow B1, .., Bn$ is True in I iff the formula $A1 \wedge ... \wedge Am \Rightarrow B1 \vee ... \vee Bn$ is True in I.
That is, $A1, .., Am \rightarrow B1, .., Bn$ is

- falsifiable iff there exists I which makes $A1 \wedge ... \wedge Am \wedge \neg B1 \wedge ... \wedge \neg Bn$ True,
- satisfiable iff there exists I which makes $\neg A1 \vee ... \vee \neg Am \vee B1 \vee ... \vee Bn$ True.

2.1.2 Rules system

The inference rules of the system G directly reflect the semantics of the logical connectives. There are two inference rules for each connective: one operating on a formula occuring in the antecedent of the sequent, and the second operating on a formula occuring in the succedent of the sequent .

$$\frac{A, B, \Gamma \rightarrow \Delta}{A \wedge B, \Gamma \rightarrow \Delta} \qquad \frac{\Gamma \rightarrow \Delta, A \quad \Gamma \rightarrow \Delta, B}{\Gamma \rightarrow \Delta, A \wedge B}$$

$$\frac{A,\Gamma\to\Delta \quad B,\Gamma\to\Delta}{A\vee B,\Gamma\to\Delta} \qquad \frac{\Gamma\to\Delta,A,B}{\Gamma\to\Delta,A\vee B}$$

$$\frac{\Gamma\to\Delta,A \quad B,\Gamma\to\Delta}{A\Rightarrow B,\Gamma\to\Delta} \qquad \frac{A,\Gamma\to\Delta,B}{\Gamma\to\Delta,A\Rightarrow B}$$

$$\frac{\Gamma\to\Delta,A}{\neg A,\Gamma\to\Delta} \qquad \frac{A,\Gamma\to\Delta}{\Gamma\to\Delta,\neg A}$$

$$\frac{A[t/x],\forall xA,\Gamma\to\Delta}{\forall xA,\Gamma\to\Delta} \qquad \frac{\Gamma\to\Delta,A[y/x]}{\Gamma\to\Delta,\forall xA}$$

$$\frac{A[y/x],\Gamma\to\Delta}{\exists xA,\Gamma\to\Delta} \qquad \frac{\Gamma\to\Delta,A[t/x],\exists xA}{\Gamma\to\Delta,\exists xA}$$

where y (called *eigenvariable*) does not occur free in the lower sequent,
and t is any term free for x in A.

2.1.3 Axioms and deduction tree

Definition : An *axiom* is any sequent $\Gamma \longrightarrow \Delta$ such that Γ and Δ contain some common formula.

Property : Every axiom is valid.

Definition : A *deduction tree* for $\Gamma \longrightarrow \Delta$ is a tree where :

- the root is labeled by $\Gamma \longrightarrow \Delta$
- we get children nodes from a parent node using an inference rule. The "parent node" is labeled with an instance of the conclusion of the rule, and the "children" nodes are labeled with the corresponding instances of the premises .
- a *leaf* is labeled with a sequent $\Gamma \longrightarrow \Delta$ where:
  - either $\Gamma \longrightarrow \Delta$ is an axiom,
  - or Γ and Δ are disjoint sets of atomic formulas.

Definition : A *proof tree* is a finite deduction tree, where all leaves are labeled with axioms.

Definition: A *counter-example tree* is a deduction tree which is not a proof tree, that is:
- either there is a non-axiom leaf,
- or there is an infinite branch.

Example : here is a deduction tree for the formula $F \equiv (P \Rightarrow Q) \Rightarrow (\neg Q \Rightarrow \neg P)$

$$\begin{tikzcd}
& \to (P\Rightarrow Q)\Rightarrow(\neg Q\Rightarrow\neg P) \arrow[d, dash] & \\
& P\Rightarrow Q\to\neg Q\Rightarrow\neg P \arrow[dl, dash] \arrow[dr, dash] & \\
\to P,\neg Q\Rightarrow\neg P \arrow[d, dash] & & Q\to\neg Q\Rightarrow\neg P \arrow[d, dash] \\
\neg Q\to P,\neg P \arrow[d, dash] & & Q,\neg Q\to\neg P \arrow[d, dash] \\
\neg Q,P\to P & & Q\to\neg P,Q
\end{tikzcd}$$

It is a proof tree.

2.2 Search procedure

We now describe an algorithm to construct a deduction tree for a sequent in a systematic fashion. Contrary to the algorithm defined in [Gallier, 86], this one builds the tree in a depth-first fashion. The procedure *develop* is inductive . It takes a sequent as an input, and gives a deduction tree as a result.

```
PROC develop ( sequent, deduction-tree )
BEGIN
IF leaf( sequent )
|  THEN deduction-tree <- sequent
|  ELSE
|  list-of-children <- apply-a-rule(sequent)
|  child <- first-member (list-of-children)
|  DO
|  |    develop (child, child-tree)
|  |    child <- succedent(list-of-children)
|  UNTIL no-more-children
|
|                                 sequent
|  deduction-tree <-           /     |     \
|                  child-tree-1 ...  child-tree-n
FI
END
```

Note : *apply-a-rule(sequent)* is a function searching for an inference rule to apply on *sequent*; it gives as a result the sequents resulting from applying the rule to *sequent*. Generally, more than one rule can be applied, and the choice of the rule to apply may be guided by heuristics.

Definition : $\Gamma \to \Delta$ is provable by the system G iff *develop* ($\Gamma \to \Delta$) produces a proof tree.

Properties : the procedure is

- *sound* : if $\Gamma \to \Delta$ is provable then $\Gamma \to \Delta$ is valid,
- *complete* : if $\Gamma \to \Delta$ is valid then $\Gamma \to \Delta$ is provable, providing, for the First-Order Calculus, that the procedure *apply-a-rule* is fair,
- *decidable* for the propositional calculus.

3 Modifying the system G

We now describe two optimisations for the system G. The first one concerns an extension of the rule system, and the second one concerns a modification of the search procedure. Both optimisations use a sequent property we call "sequents subsumption". We first define this property.

Definition : a sequent $\Gamma \to \Delta$ *subsumes* a sequent $\Gamma_1 \to \Delta_1$ **iff** $\Gamma \subseteq \Gamma_1$ and $\Delta \subseteq \Delta_1$.

Property 1: $\Gamma \to \Delta$ subsumes $\Gamma_1,\Gamma \to \Delta\ ,\Delta_1$.

Property 2: if $\Gamma \to \Delta$ subsumes $\Gamma_1 \to \Delta_1$ **then if** $I \models \Gamma \to \Delta$ **then** $I \models \Gamma_1 \to \Delta_1$.

Property 3: if $\Gamma_1 \to \Delta_1$ subsumes $\Gamma \to \Delta$ **then if** $\Gamma_1 \to \Delta_1$ is provable **then** $\Gamma \to \Delta$ is provable and any proof of $\Gamma_1 \to \Delta_1$ is a proof for $\Gamma \to \Delta$.

3.1 Construction of macro-rules

The modification of the system proposed here concerns the system of inference rules. In the system G, the rules system is closed. It describes the semantics of the primitive connectives: $\wedge, \vee, \Rightarrow, \neg, \exists, \forall$. So, in order to be handled by the system G, any formula must be expressed in

terms of these primitive connectives, which will then be successively dealt with, by the appropriate rules. However, if a formula is to express a significant relationship between its subformulas, stating it using primitive connectives may be rather unnatural (cf the "relationships" *equivalent, exclusive-or, n-among-m,* etc...). We call such structures *macro-connectives.* We would like the system to be able to directly deal with them through associated *macro-rules.* The semantics of such a structure will be explicitly rendered by the corresponding *macro-rule.* Moreover, it will be accounted for, in a proof, by a single derivation step.
The method described in this section is restricted to the handling of propositional macro-connectives.

3.1.1 Definitions

A *macro-connective* with arity n is a connective defined from an application of primitive connectives to n formulas.
Consider for instance the binary macro-connective $\Leftrightarrow$ defined by

$$A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$$

Accordingly, a *macro-rule* is an inference rule for a macro-connective. It expresses the semantics of the macro-connective with respect to the semantics of the n arguments formulas.

3.1.2 A method to construct a macro-rule

Similarly to primitive connectives, each macro-connective gives rise to two macro-rules. One deals with an occurrence of the macro-connective in the left part of the sequent, and the other deals with an occurrence of the macro-connective in the right part of the sequent. A macro-rule is produced by synthetizing the deduction tree of the macro-connective.
The deduction tree is generated by successively applying the inference rules to the primitive connectives involved in the macro-connective, so as to completely develop the structure. This deduction tree is then pruned by discarding all axiom leaves and *redundant* leaves. The resulting macro-rule consists in the leaves of this pruned tree as its premises, and the macro-connective structure as its conclusion.

Definition : A sequent S is *redundant* with respect to a set E of sequents **iff** S is implied by E . That is, taking E $= S_1, \cdots, S_n$, S is redundant with respect to E **iff** for all I, if I $\models \bigwedge_i S_i$ then I $\models$ S.

Proposition : A sequent S is redundant with respect to a set E of *leaf sequents* **iff** there exists a sequent S_n such as S_n follows from E by *transitivity*, and S_n subsumes S.

Definition : *rule of transitivity on sequents*
Given two sequents S_1 and S_2, if there exists four finite sets of formulas $\Gamma, \Delta, \Gamma_1, \Delta_1$ and a formula F such as $S_1 \equiv \Gamma \to F, \Delta$ and $S_2 \equiv F, \Gamma_1 \to \Delta_1$, then S_1 and S_2 imply $\Gamma, \Gamma_1 \to \Delta, \Delta_1$ by transitivity.

Definition : S *follows from E by transitivity* iff there exists $S_1, ..., S_n$ such that

- $S_1 \in$ E
- for $i = 1...n-1$, there exists Seq in E $\cup\{S_j, j \leq i\}$ such that S_i and Seq imply S_{i+1} by transitivity
- $S_n \equiv$ S

Note: the *rule of transitivity* is based on a well-known semantical result:

- first, this result is used to defined the *Cut rule* for the Gentzen Sequent Calculus LK (this rule formalizes the use of an auxiliary lemma in a proof,[Gallier, 86] p109)
- then, this result is used to defined the *rule of Resolution.* The similarity is easy to detect if we refer to the semantical definition of a sequent : I $\models A1, .., Am \to B1, .., Bn$ iff I $\models A1 \wedge ... \wedge Am \Rightarrow B1 \vee ... \vee Bn$, that is iff I $\models \neg A1 \vee ... \vee \neg Am \vee B1 \vee ... \vee Bn$. The clausal form is an instance of the latter form, and the rule of Resolution is the same as the rule of transitivity applied on formulas in clausal form.

3.1.3 Example

Given the macro-connective $\Leftrightarrow$ defined by $A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$, there are two macro-rules asociated with it: "$\Leftrightarrow\to$" and "$\to\Leftrightarrow$" .
1 - **construction of "$\Leftrightarrow\to$"**
deduction tree for "$\Leftrightarrow\to$ ":

$$\begin{tikzcd}[column sep=tiny]
 & & & A \Leftrightarrow B \to \arrow[d, dash] & & & \\
 & & & (A \Rightarrow B) \wedge (B \Rightarrow A) \to \arrow[d, dash] & & & \\
 & & & A \Rightarrow B, B \Rightarrow A \to \arrow[dll, dash] \arrow[drr, dash] & & & \\
 & B \Rightarrow A \to A \arrow[dl, dash] \arrow[dr, dash] & & & & B, B \Rightarrow A \to \arrow[dl, dash] \arrow[dr, dash] & \\
\to A, B & & A \to A & & B \to B & & B, A \to
\end{tikzcd}$$

synthetized tree:

$$\begin{tikzcd}[column sep=tiny]
 & A \Leftrightarrow B \to \arrow[dl, dash] \arrow[dr, dash] & \\
\to A, B & & A, B \to
\end{tikzcd}$$

inference rule :

$$\frac{\Gamma \to \Delta, A, B \quad A, B, \Gamma \to \Delta}{A \Leftrightarrow B, \Gamma \to \Delta}$$

2 - **construction of "$\to\Leftrightarrow$"**
deduction tree for "$\to\Leftrightarrow$":

$$\begin{tikzcd}[column sep=tiny]
 & \to A \Leftrightarrow B \arrow[d, dash] & \\
 & \to (A \Rightarrow B) \wedge (B \Rightarrow A) \arrow[dl, dash] \arrow[dr, dash] & \\
\to A \Rightarrow B \arrow[d, dash] & & \to B \Rightarrow A \arrow[d, dash] \\
A \to B & & B \to A
\end{tikzcd}$$

synthetized tree:

$$\begin{tikzcd}[column sep=tiny]
 & \to A \Leftrightarrow B \arrow[dl, dash] \arrow[dr, dash] & \\
A \to B & & B \to A
\end{tikzcd}$$

inference rule :

$$\frac{A, \Gamma \to \Delta, B \quad B, \Gamma \to \Delta, A}{\Gamma \to \Delta, A \Leftrightarrow B}$$

3.1.4 Some significant rules

Of special interest are the macro-rules for n-ary connectives "1 among n" (noted **XORn** and defined by $(A_1 \vee ... \vee A_n) \wedge \neg(A_1 \wedge A_2) \wedge ... \wedge \neg(A_{n-1} \wedge A_n)$), and "equivalence between n" (noted **EQUIn** and defined by

$(A_1 \wedge ... \wedge A_n) \vee (\neg A_1 \wedge ... \wedge \neg A_n))$. Indeed, those connectives are quite useful to describe some "combinatorial" problems, and introducing those connectives in usual deduction systems (including *resolution* methods, and the "standard" system G) is costly in time and space. We establish the general rules to deal with them directly (cf figure 1).

Remarks

-**XORn**: use of XOR3 by different systems of resolution.

- with the "standard" system G: the deduction of $XOR3(A, B, C) \rightarrow$ from the formula $(A \vee B \vee C) \wedge \neg(A \wedge B) \wedge \neg(B \wedge C) \wedge \neg(A \wedge C)$ gives a deduction tree whose depth is 10, and has 19 nodes and 15 leaves, 9 of them are axioms.
- with the system G including the macro-rule XOR3: the deduction of $XOR3(A, B, C) \rightarrow$ gives a deduction whose depth is 1, and has 1 node and 3 leaves.

-**EQUIn**: end of construction of "$\rightarrow EQUIn$"

- synthetized tree before using the rule of transitivity (the axiom leaves and the subsumed leaves only have already been removed):

$$\begin{tikzcd}[column sep=small]
 & & & \rightarrow EQUIn(I_1, ..., I_n) \arrow[dlll, dash] \arrow[dl, dash] \arrow[dr, dash] \arrow[drrr, dash] & & & \\
I_1 \rightarrow I_2 & ... & I_1 \rightarrow I_n & ... & I_n \rightarrow I_1 & ... & I_n \rightarrow I_{n-1}
\end{tikzcd}$$

It is a tree with $n * (n-1)$ leaves.

- synthetized tree after using the rule of transitivity:

$$\begin{tikzcd}[column sep=small]
 & & & \rightarrow EQUIn(I_1, ..., I_n) \arrow[dlll, dash] \arrow[dl, dash] \arrow[d, dash] \arrow[drrr, dash] & & & \\
I_1 \rightarrow I_2 & ... & I_{i-1} \rightarrow I_i & I_i \rightarrow I_{i-1} & & ... & I_n \rightarrow I_{n-1}
\end{tikzcd}$$

It is a tree with $2 * (n-1)$ leaves.

3.1.5 Conclusion

In our opinion, an interesting point of the macro-rule notion is to propose a mean to generate automatically new rules, when they appear useful in some application domain, instead of giving a new system of rules obtained from adding an exhaustive collection of new inferences rules. Indeed, this system would be frozen, as the first one, and there surely would exist some "non universal relationship", not taken into account by that system, and although very useful in some special domain. So, the idea is to set up a dedicated system for each application domain. The global result is to have, for each domain, a system of rules with all and only useful rules (to limit the waste of time in searching for a rule to apply). The construction of macro-rules is not costly (rules are computed once for all) and the use of those rules can give an important reduction of the depth of the proof tree (by saving time and space in deductions).

3.2 Alternative cuts

The second optimisation we have implemented with the system G is about the search procedure. It consists to detect dynamically whether useless steps are being performed in the development, and if so, to prune pending alternative branches subordinate to those steps.

3.2.1 Principle

Consider the following derivation of the sequent $S \equiv A \vee B, C \vee D \rightarrow C, D$

$$\begin{tikzcd}[column sep=small]
 & S \equiv A \vee B, C \vee D \rightarrow C, D \arrow[dl, dash] \arrow[dr, dash, "\times" description] & \\
S_1 \equiv A, C \vee D \rightarrow C, D \arrow[d, dash] \arrow[dr, dash] & & S_2 \equiv B, C \vee D \rightarrow D \\
S_{11} \equiv A, C \rightarrow C, D & S_{12} \equiv A, D \rightarrow C, D &
\end{tikzcd}$$

S1 is provable, and the proof of S1 does not rely on the subformula A (which is the only formula from S1 not being in S already). So, the sub-sequent $C \vee D \rightarrow C, D$ from S1 is provable. As this sub-sequent comes from S without being modified, we conclude that S could be proved without developing the formula $A \vee B$. That is:

- S is proved, and its proof is the proof of $C \vee D \rightarrow C, D$, namely, the proof of S1.

- developing $A \vee B$ is a useless step, so this development can be stopped.

- the pending alternative S2, as it comes from the development of $A \vee B$, is suppressed.

The resulting proof is the following one:

$$\begin{tikzcd}[column sep=small]
 & A \vee B, C \vee D \rightarrow C, D \arrow[dl, dash] \arrow[dr, dash] & \\
A \vee B, C \rightarrow C, D & & A \vee B, D \rightarrow C, D
\end{tikzcd}$$

3.2.2 Method

Consider a sequent S, in which we decide to develop the formula F_s using the rule R. We note $S \equiv F_s + Seq$, where Seq is the "skeleton sequent", that is, the sequent S without the selected formula F_s.

The application of the inference rule R to S can be schematized by the following figure:

$$\begin{tikzcd}[column sep=small]
 & F_s + Seq \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\
\cdots FL_{child_i} + Seq \arrow[d] & FL_{child_{i+1}} + Seq \cdots & FL_{child_n} + Seq \\
develop\ (Seq_{child_i}) & &
\end{tikzcd}$$

$\rightarrow deduction - tree_{child_i}$

$\rightarrow UF_{child_i}$

where

- FL_{child_j} is the list of the subformulas of F_s developed by the jth premise of the inference rule R,
- Seq is the "skeleton sequent" inherited from the parent sequent S. It contains every formula of S not modified by the application of the rule,
- $Seq_{child_j} \equiv FL_{child_j} + Seq$. Seq_{child_j} is the jth child resulting from applying the rule,
- UF_{child_j} is the set of formulas from Seq_{child_j} effectively used to prove Seq_{child_j}.

$$\frac{A_1, A_2, \ldots, A_n, \Gamma \rightarrow \Delta \qquad \Gamma \rightarrow \Delta\,,\, A_1, A_2, \ldots, A_n}{EQUIn(A_1, ..., A_n),\, \Gamma \rightarrow \Delta}$$

$$\frac{A_1, \Gamma \rightarrow \Delta\,, A_2 \quad A_2, \Gamma \rightarrow \Delta\,, A_1 \;\ldots\; A_{i-1}, \Gamma \rightarrow \Delta\,, A_i \quad A_i, \Gamma \rightarrow \Delta\,, A_{i-1} \;\ldots\; A_{n-1}, \Gamma \rightarrow \Delta\,, A_n \quad A_n, \Gamma \rightarrow \Delta\,, A_{n-1}}{\Gamma \rightarrow \Delta\,,\, EQUIn(A_1, ..., A_n)}$$

$$\frac{A_1, \Gamma \rightarrow \Delta\,,\, A_2, \ldots, A_n \quad A_2, \Gamma \rightarrow \Delta\,,\, A_1, A_3, \ldots, A_n \quad \ldots \quad A_n, \Gamma \rightarrow \Delta\,,\, A_1, \ldots, A_{n-1}}{XORn(A_1, ..., A_n), \Gamma \rightarrow \Delta}$$

$$\frac{\Gamma \rightarrow \Delta\,,\, A_1, \ldots, A_n \quad A_1, A_2, \Gamma \rightarrow \Delta \quad \ldots \quad A_{n-1}, A_n, \Gamma \rightarrow \Delta}{\Gamma \rightarrow \Delta\,,\, XORn(A_1, ..., A_n)}$$

Figure 1: Generic rules for EQUIn and XORn

The cut:

- *Cut condition* :
  Seq_{child_i} is provable and $FL_{child_i} \cap UF_{child_i} = \emptyset$
- *Cut actions* :
  - suppress from the pending alternative list the child sequents Seq_{child_j} not developed yet ,
  - note that the formula F_s is useless to prove S,
  - give as a proof for S the proof of Seq_{child_i}.
- *Cut justification :*
  When the cut condition is verified the current situation is the following one: the sequent Seq_{child_i} is provable and its proof is independent of the formulas in FL_{child_i}; hence the proof only depends on the "skeleton sequent" Seq. That is, Seq is provable. Seq subsumes $F_s + Seq$ (according to property 1 on subsumbtion), and Seq is provable; hence, according to property 3, S is provable, and the proof of Seq_{child_i} is a proof for S.

3.2.3 New search procedure

This new search procedure implements the *cut*. It takes a sequent as an input and gives as a result both the deduction tree for the sequent, and the list of formulas of the sequent necessary for the proof (when the deduction tree is a proof tree).

```
PROC develop (sequent,deduction-tree,UF-parent)
BEGIN
IF leaf( sequent )
|  THEN deduction-tree <- sequent
|     IF axiom (sequent)
|      THEN  UF-parent <- "both formulas which
|                          displayed the axiom"
|      ELSE  UF-parent <- "all  formulas
|                          of the sequent"
|
|  ELSE
|   list-of-children <- apply-a-rule(sequent)
|   child <- first-member (list-of-children)
|   DO
|   |  develop (child, child-tree,UF-child)
|   |  IF cut-condition (child,UF-child)
|   |             GO-TO-END sibling-cut
|   |  child <- succedent (list-of-children)
|   UNTIL  sibling-cut OR no-more-children
|
|    END sibling-cut :
|
|     deduction-tree <-  child-tree
|     UF-parent <-  UF-child
|
|    END no-more-children :
|                                  sequent
|     deduction-tree <-          /    |    \
|                  child-tree-1 ...  child-tree-n
|
|     UF-parent <- construct(Ui UF-child-i,sequent)
|
FI
END
```

where *construct*($\bigcup_i UF_{child_i}$, *sequent*) $\equiv$

$$[\bigcup_j (UF_{child_j} \setminus FL_{child_j})] \bigcup F_s$$

where F_s is the formula being developed in *sequent*; *construct* is used to mention that F_s is used in the proof of *sequent*.

Note: the test *sibling-cut* to go out of the loop is realised before the other test *no-more-children* is. This enables one to supress a proof step, even when the detection of uselessness is only made at the end of the step. In this case, the immediate effect of the detection is the reduction of the proof given as a result for this node. The postponed effect is to make it possible to detect some higher cuts, which could not be detected otherwise.

3.3 Example

Here is an example to illustrate the use of the *cut*. Consider the sequent S such that
$S \equiv XOR3(B, A \vee Q, C), P \Leftrightarrow Q \rightarrow A, P \Rightarrow Q$. The figure 2 shows the deduction tree of S processed using the "standard" procedure *develop*. Using the procedure *develop* with cut implementation, the two sub-trees $sub - tree_{child_2}$ and $sub - tree_{child_3}$ are not expanded and the resulting proof is the following:

$$\begin{array}{c} xor3(B, A \vee Q, C), P \Leftrightarrow Q \rightarrow A, P \Rightarrow Q \\ | \\ xor3(B, A \vee Q, C), P \Leftrightarrow Q, P \rightarrow A, Q \\ / \quad \backslash \\ xor3(B, A\vee Q,\ C), P, Q, P \rightarrow A, Q \quad xor3(B, A\vee Q,\ C), P \rightarrow A, Q, P, Q \end{array}$$

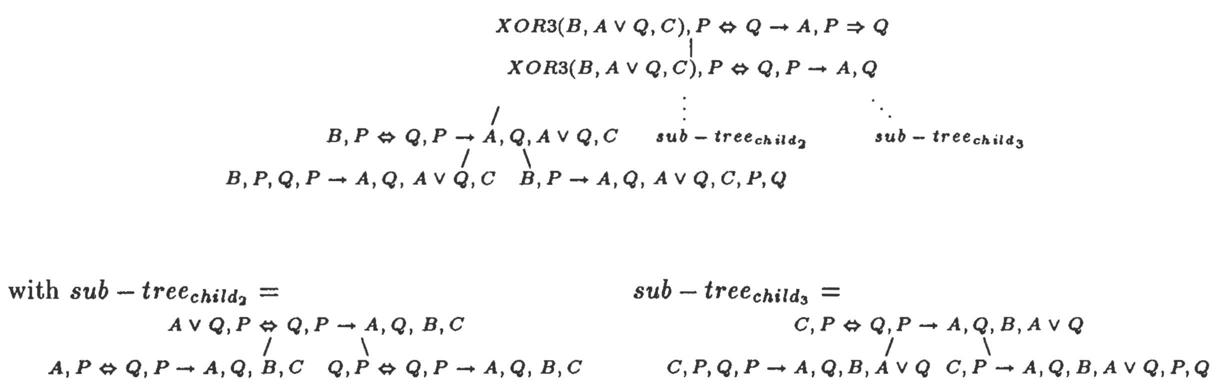

XOR3(B, A ∨ Q, C), P ⇔ Q → A, P ⇒ Q
XOR3(B, A ∨ Q, C), P ⇔ Q, P → A, Q
B, P ⇔ Q, P → A, Q, A ∨ Q, C
sub − tree_child2
sub − tree_child3
B, P, Q, P → A, Q, A ∨ Q, C
B, P → A, Q, A ∨ Q, C, P, Q
with sub − tree_child2 =
A ∨ Q, P ⇔ Q, P → A, Q, B, C
A, P ⇔ Q, P → A, Q, B, C
Q, P ⇔ Q, P → A, Q, B, C
sub − tree_child3 =
C, P ⇔ Q, P → A, Q, B, A ∨ Q
C, P, Q, P → A, Q, B, A ∨ Q
C, P → A, Q, B, A ∨ Q, P, Q

Figure 2: Deduction tree of the sequent S according to the system G

3.4 Some results

Some tests have been performed to compare the standard search procedure (p1) and the new one with cut (p2). For guidance, we give the computional time for a few combinatorial problems: the Andrew's Challenge [Pelletier, 86] and the lemma of Schur [Schur, 16] performed with respectively p1 and p2:

Andrew	324s	385s
Schur5	82s	no result
Schur4	30s	2200s

3.5 Conclusion

A step is regarded as useless iff there exists one (or more) child from the step whose proof is free from the fulfilment of the step. If so, all the pending alternative childs are pruned, and the useless step is removed from the resulting proof. Hence, this yields a reduction in both the effective proof (saving the time and space needed to develop the alternative branches removed), and the resulting proof (giving a more natural proof, with only meaningfull steps). In fact, the gain may be all the more significant as the arity of the connective expanded by the step is important. Our method is reminiscent of that given by [Oppacher and Suen, 86], but the latter is about semantic tableaux, and deals with binary trees.

4 Conclusion

This paper presents improvements of the system G, a Sequent Calculus of Gentzen. Both proposed extensions lead to more efficiency and naturalness of the proofs. Actually, both methods are examples of partial evaluation ([van Harmelen and Bundy, 88]) applied to the procedure *develop*. In so far as the first one consists in performing once for all the computing of a special sequence of connectives, and the second one applies the principle which consists in immediately declaring as true any sequent that includes a true sequent. The methods have been implemented in Prolog-MALI [Brisset, 89]. The resulting system is considered as the basis for the specification of an adaptative prover. Our project is to produce automatically a set of macro-rules for a specific domain, given a sample of theorems on this domain. The realisation of this project contains an extension of the macro-rule notion. Actually, it supposes a macro-rules to be able to deal not only with a connective, but also with "macro structures"; that is to deal directly with a combination of primitive connectives not labeled explicitly with a macro-connective name. Furthermore the notion of macro-connectives could be extended to structures handling more than one formula in a sequent. With this broader perspective, macro-connectives may be view as a means to syntactically characterize some semantical features of a given domain.

References

[Bowen, 82] K.A. Bowen. Programming with Full First-Order Logic. *Machine Intelligence*, 10:421–440, 1982.

[Brisset, 89] P. Brisset. *Implémentation d'un langage de programmation logique d'ordre supérieur avec MALI.* Rapport de Recherche 1119, INRIA, 1989.

[Chang and Lee, 73] Chang and Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, New York and London, 1973.

[Gallier, 86] J.H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving.* Harper and Row, New York, 1986.

[Oppacher and Suen, 86] F.Oppacher and E.Suen. Controlling Deduction with Proof Condensation and Heuristics. *Proceeding of the 8th International Conference on Automated Deduction*, 384–393, 1986.

[Pelletier, 86] F.J. Pelletier. Seventy-Five Problems for Testing ATP. *Journal of Automated Reasoning*, 2:191–216, 1986.

[Schur, 16] I.Schur. Uber Die Kongruenz Xm+Ym = Zm mod(p). *Jber Deutsch Verein*, 25:114–116, 1916.

[Smullyan, 68] R.M. Smullyan. *First Order Logic.* Springer Verlag, Berlin, 1968.

[Stickel, 88] M.E. Stickel. Resolution Theorem Proving. *Annual Review Computer Science*, 285–316, 1988.

[van Harmelen and Bundy, 88] F. van Harmelen and A. Bundy. Explanation-Based Generalisation = Partial Evaluation. *Artificial Intelligence*, 36:401–412, 1988.

Improved Relaxation and Search Methods for Approximate Constraint Satisfaction with a Maximin Criterion*

Paul Snow
Computer Science Department
Plymouth State College
Plymouth, NH 03264 USA

Eugene C. Freuder
Computer Science Department
University of New Hampshire
Durham, NH 03824 USA

Abstract

Rosenfeld et al. [1976] presented a relaxation method for approximate constraint satisfaction to be used with a "fuzzy logic" conception of the merit of a labeling. We present a modification of the Rosenfeld et al. result in which each candidate label initiates revision of the estimated desirability of related labels at neighboring nodes at most once. The candidate label is then retired from further analysis. We also present a result that aids in the search for an actual labeling once the relaxation is quiescent. Finally, we note that these methods can be extended to approximate constraint satisfaction outside of "fuzzy logic" conceptions. In particular, we discuss the notion of an "undominated" labeling. The relaxation and search results presented here are easily adapted to the search for an undominated labeling.

1 Introduction

In the most familiar kind of constraint satisfaction problem [Haralick *et al.*, 1978; Dechter and Pearl, 1987], the interacting interpretations ("labels") for the various objects under discussion are either compatible in certain combinations, or they are not. The goal is to find at least one set of interpretations, with exactly one interpretation for each object (a "labeling"), in which all the interpretations are compatible with one another.

The purpose of this paper is to present methods for problems where compatibility is not simply a yes or no predicate. That is, we consider problems where there are degrees to which a constraint can be satisfied.

Such problems arise in several ways. Perhaps we feel that some combination of labels is more likely than another, although both are possible. Waltz [1975] encountered such a situation in his work on line drawings with shadows. Perhaps the constraints themselves are not clear-cut. For example, maybe some numeric label must be approximately equal, but not necessarily exactly equal, to another. Yager [1988] discusses such problems in the context of fuzzy decision theory. Or perhaps we simply prefer one set of interpretations over another. Rosenfeld et al. [1976] consider cases where we prefer to find convex objects in a scene if it is possible that any are there.

2 Notation and Preliminaries

We conceive of our problem as one of finding labels for the nodes of a graph subject to binary constraints governing labels at adjacent nodes. We shall call the set of nodes in the graph N.

For each node n in N, there is a set of candidate labels. A particular candidate label is denoted by the tuple (n, c), where n is the node in N and c distinguishes among the candidate labels available at n.

To express our approximate constraints, we introduce a "goodness" function, $g(n, c, m, d)$, where (n, c) and (m, d) are candidate labels at adjacent nodes n and m. For convenience, we bound $g(,,,)$ above and below, confining it to values in the closed unit interval. If $g(n, c, m, d)$ is zero, then (n, c) and (m, d) are incompatible in the usual sense: no labeling containing both of them is acceptable. Otherwise, a higher goodness value is better than a lower one, all other things being equal.

The nature and source of the goodness values are intentionally left unspecified. They may reflect expert judgments of fuzzy membership grades in the satisfaction sets of various constraints, estimates of likelihood, or expressions of preference.

3 The Merit of a Labeling

We assume that we have no other information about the desirability of different labelings except the edgewise goodness introduced above. Despite this ignorance, we wish to devise a figure of merit for a labeling as a whole, to guide us in choosing a particular labeling.

Many conventions are possible. The convention that will be pursued here is

$$M(L) = \min g(n, c, m, d)$$

where $M(L)$ is the merit for labeling L, and the minimum is taken over all labels that appear at adjacent nodes in L. In choosing a labeling, therefore, we shall seek to maximize the merit, or "maximin" the goodness of adjacent label pairs.

*This material is based in part upon work supported by the National Science Foundation under Grant No. IRI-8601209 and Grant No. IRI-8913040.

While our choice is hardly the only possibility, it comports well with several defensible notions about what a useful figure of merit should be. Rosenfeld, et al. [1976] and Yager [1988] arrived at the same figure of merit using a fuzzy decision theory argument. If $g(,,,)$ is a fuzzy degree of constraint satisfaction, them $M()$ is the ordinary implementation of the fuzzy AND operation. That is, $M()$ is a conventional measure of the degree to which all the fuzzy edgewise constraints are satisfied simultaneously.

Interest in maximining is not restricted to fuzzy theorists. Dyson [1980] has argued for the generality of maximin in multi-criteria decision making. Or, perhaps the labeling represents the deployment of assets in a competitive environment. Opponents can often be counted on to exploit local weaknesses in such domains.

The labeling may represent an explanation of a given set of facts, with some parts of the explanation fitting the facts better that others. The hoary heuristic "a chain is as strong as its weakest link" may be a plausible predictor of how convincing the explanation is over-all.

Other motivations are possible. If $g(,,,)$ is the joint probability of the label pairs, $M()$ is an upper bound on the joint probability of the labeling as a whole. Such an upper bound might be useful in planning applications, where the whole plan's chances of success can not exceed the chances of any part of the plan.

It is worth noting that the goodness function is thought of as a description of how well adjacent labels go together *paribus ceteris*, independent of how well they suit the context of a specific problem. We believe that this is in the spirit of the more familiar, two-valued, compatibilities. This approach contrasts with that of Rosenfeld et al. [1976], who view their counterpart of goodness as an estimate of compatibility to be revised by the other estimates in a specific problem.

4 An Upper Bound on the Merit

If labeling L contains label (n, c), then it can easily be shown that

$$M(L) \leq \min \max g(n, c, m, d)$$

where the minimum is taken over all nodes m adjacent to n, and the maximum is taken over all candidate labels (m, d) at each such m. If (n, c) is included in L, then the maximization describes the best goodness which involves (n, c) that we can hope to find among the candidate labels at each adjacent node. The minimum of these maxima clearly bounds $M(L)$, which is the minimum of all goodnesses that arise from L. For brevity, we shall denote the right side of the above inequality as $B(n, c)$.

This upper bound is generally loose. We shall attempt to improve it by relaxation. First, we shall consider Rosenfeld et al.'s [1976] relaxation, and then we shall develop a variation of it. Even after relaxation, however, the refined $B(,)$ values are typically loose. Thus, we must generally search among labelings to find an actual labeling that realizes the maximum possible merit. We can exploit a result about the relaxation process to help us in that search.

5 Rosenfeld et al. Relaxation

The goodness $g(n, c, m, d)$ is itself an upper bound on the merit of any labeling containing (n, c) and (m, d). Rosenfeld et al. [1976] use $B(n, c)$ and $B(m, d)$ to derive a tighter bound than $g(n, c, m, d)$. They revise the goodness values. We find it convenient to introduce a new edgewise bounding quantity, $h(n, c, m, d)$, initially equal to the corresponding goodness, and later the subject of further refinement by relaxation.

These edgewise bounds can then be used to calculate new, tighter $B(,)$ values, which can tighten the edgewise bounds further, and so on. The relaxation algorithm that accomplishes this is:

```
begin
  for all candidate label pairs $(n, c)$ and $(m, d)$
    do $h(n, c, m, d) := g(n, c, m, d)$

  for all candidate labels $(n, c)$
    do $B(n, c) := \min \max h(n, c, m, d)$
      min and max taken as described above

  repeat
    for all candidate pairs $(n, c)$ and $(m, d)$ do
      begin
        $h(n, c, m, d) := \min( h(n, c, m, d), B(n, c),$
                                    $B(m, d) )$
        if $h(n, c, m, d)$ changes, then recompute
                                    $B(n, c)$ and $B(m, d)$
      endfor
  until no changes in any $h(,,,)$ occur during
                                    an iteration
end.
```

The idea, of course, is that $h(n, c, m, d)$, $B(n, c)$, and $B(m, d)$ all bound the merit of any labeling that contains both (n, c) and (m, d). So, any $h(,,,)$ that exceeds either of the corresponding $B(,)$'s is looser than it has to be based on other known facts.

Rosenfeld et al. show that in the case where $h(,,,)$ is two-valued (zero or unity, that is, the familiar yes or no compatibility), the quiescent values of the $B(,)$'s are isomorphic to the result of a Waltz [1975] relaxation. "Possible" labels found by Waltz have a quiescent $B(,)$ value of unity in the two-valued Rosenfeld et al.; "impossible" labels have a value of zero.

6 A Refined Relaxation

Another algorithm that depends on $B(,)$ values can be devised that relies on the same notion of bound consistency as Rosenfeld et al.. This new algorithm achieves quiescence with no candidate label's $B(,)$ being propagated to related candidates' $h(,,,)$ values at neighboring nodes more than once.

The extra bookkeeping required to accomplish this result is to track a new number. We call the new number b; it is the smallest $B(,)$ value among the labels that have not been compared to their neighbors. It is simple to confirm that no label will have a quiescent $B(,)$ smaller than the original value of b.

That is because the fundamental relaxation relation:

$$h(n,c,m,d) := \min(\ h(n,c,m,d),\ B(n,c),\ B(m,d)\ )$$

cannot lead to a decrease in $B(n,c)$ unless $B(m,d)$ is lower, which is not the case if $B(n,c)$ is already the lowest $B(,)$. Those labels whose $B(,)$'s are already equal to b will therefore not have their values further reduced by the relaxation.

The strategy, then, is to compute b, locate the labels that have b as their $B(,)$, and propagate that bound to the $h(,,,)$ values at each of their neighbors. Any labels that thereby fall to a $B(,)$ of b are similarly treated, until all the labels that will be pulled down to b have been identified and their effect has been propagated.

Once the original b has been fully propagated, we recompute the $B(,)$'s for labels that have been revised, but not all the way to b. Then, we are ready to compute a new b among all labels whose $B(,)$ is not the original b. Labels that have already been relaxed to the old b will not be relaxed again, and the effects of their presence is already reflected in the current $h(,,,)$ values at adjacent nodes. We can therefore exclude them from any further analysis. Among the remaining labels, we proceed with the new b much the same way as we did before with the original b.

We thus perform one round of relaxation for each distinct value of b we encounter. The results of each round are isomorphic to those of a classic Waltz relaxation. The candidate labels are divided into two classes, those with $B(,)$'s equal to b, and those with higher $B(,)$'s. The basic mechanism of each round's relaxation is also very similar to the Waltz procedure, except we emphasize the "zeros" rather than the "ones", and we do not throw labels away, but rather record their $B(,)$'s and associated $h(,,,)$'s.

To know when we are done, we can look at another quantity we shall call t, the smallest of those $B(,)$ values that are the highest $B(,)$'s among labels at their nodes. Since some label must be included from each node in any labeling, t is an upper bound on the merit of a labeling built from the available candidates. Hence, when b equals t, we can simply set any $h(,,,)$'s which are higher than t to be equal to t and quit.

The algorithm can be specified in pseudocode as follows. We shall use a stack, Check, for the labels to be compared to their neighbors. Labels that must have their $B(,)$'s recomputed will be "marked" as such.

```
begin
  Store the $h(,,,)$'s and compute the $B(,)$'s as before.
  Compute $t$ and $b$ as described above.
  All labels are initial unmarked.

  while $t > b$ do
  begin
    for all labels do
      if $B(n,c) = b$, then push $(n,c)$ onto Check

    repeat
      pop a label $(n,c)$ from Check;
      for all $(m,d)$ where $m$ is adjacent to $n$ do
        begin
          $h(n,c,n,d) := \min(\ h(n,c,m,d),\ b\ )$;
          if there is no $(n,c')$ with $h(n,c',m,d) > b$
            then
            begin
              unmark $(m,d)$
              push $(m,d)$ onto Check
            end
            else mark $(m,d)$
        endfor
    until Check is empty

    for all marked labels $(m,d)$ do
      begin
        recompute $B(m,d)$
        unmark $(m,d)$
      endfor

    for all labels where $B(,) > b$ do
      recompute $b$ and $t$
  endwhile

  for all adjacent pairs where $h(,,,) > t$ do
    reset $h(,,,) = t$

end.
```

7 Search After Relaxation

After relaxation, we have refined $h(,,,)$ values for each adjacent pair of candidate labels. Rosenfeld et al. recommended a "best first" search among labels. They take those labels whose $B(,)$ values are the highest (i.e., our t), and see whether a labeling can be constructed using just those labels. If so, a maximin labeling has been found; introducing other labels with lower $B(,)$'s cannot improve the merit over that of the labeling found. If no labeling is found among the best labels, then the next best labels (based on $B(,)$ values) are added to the search space, and so on until a labeling is found, or until there are no new labels to try.

During each round of search, Rosenfeld et al. proceed by a simple backtracking strategy. They choose any labels at any node with more than one candidate in the current search space as the labels for that node. They then do a Waltz relaxation, and pursue the search as far as possible by instantiating another node and so on.

We consider a somewhat different search strategy. It, too, is "best first", but the search space consists of label pairs (henceforth called "links" for brevity). The links are selected for their quiescent $h(,,,)$ values, with the highest value (i.e., t) taken on the first round, and successively lower values brought in during later rounds. On each round, all links that share a single common $h(,,,)$ value are added to the search space.

Among the links added to the search space on each round, there will be at least one link whose $h(,,,)$ is equal to its original goodness. (This can be shown by a simple induction argument starting from the class of links whose quiescent $h(,,,)$ is the original b and working up.) Let us call such a link an "obligatory link".

Further, if there is a labeling that can be built from among the links in the current search space, that labeling will include at least one obligatory link whose goodness is the lowest among the links in the search space. (Suppose that on the first round, a labeling could be constructed without an obligatory link. Then such a labeling would have a higher merit than t, contrary to t being an upper bound on the merit. The argument for later rounds is similar.)

This fact can be exploited in search. We need only try to build a labeling starting with each of the obligatory links in turn. For each trial, we can employ any backtracking search strategy to find a labeling. When we have exhausted the obligatory links, however, we need not try to build on any other links; we simply bring in the next-best set of links.

As in the Rosenfeld et al. search, we can quit as soon as we find any labeling, knowing that we have achieved the highest merit possible.

8 Dominance

In some domains, we may prefer one labeling to another, even if they both have the same merit. One way that can happen is *simple dominance*. We say labeling K simply dominates labeling L if for every edge, the goodness of K is at least as great as the goodness of L, and for at least one edge, the inequality is strict.

A related standard is *lexicographic dominance*. Labeling K lexicographically dominates labeling L, if for all i between one and the number of edges, the i-th best edge of K is at least as great as the i-th best edge of L, and the inequality is strict for at least one value of i.

Clearly, simple dominance implies lexicographic dominance (but not the converse). We may wish to adopt as our labeling selection criterion that the selected labeling be lexicographically undominated (i.e. no other labeling lexicographically dominates the one selected). Such a criterion will always select a labeling that simply dominates all others if one exists (since lexicographic undominated implies simply undominated) and, of course, one that lexicographically dominates all others if one exists.

Searching for a lexicographically undominated labeling requires only a simple modification of the methods presented for maximin goodness. This makes sense, since higher (or equal) maximin goodness is implied by lexicographic dominance.

No change is required in the relaxation step, nor in the choice of best first $h(,,,)$ search, nor in the use of obligatory links. A change that is useful is that once an obligatory link is selected for trial, the attempt to build on that link should proceed in best first goodness order on the remaining links.

Generally, if a labeling is found, search must continue until all obligatory links in the current search space have been tried. Also, within a search trial based on any obligatory link, "ties" in the best first search must be pursued until and unless an i-th best link is bettered by the i-th best link of the labeling already found. Once a lexicographically undominated labeling in any current search space is found, however, no expansion of the search space is necessary. Any other labelings in an expanded search space will have lower merit than, and hence cannot lexicographically dominate, the labeling already found.

9 Conclusion

Maximining provides a simple way to combine local degrees of constraint satisfaction into a figure of merit for a labeling as a whole. This figure of merit is meaningful under a variety of notions about what constitutes a "better" labeling. Well-understood and simple-to-implement strategies are available for searches informed by the maximin criterion. With straightforward modifications, these strategies also support the stronger criterion of lexicographic undominatedness.

We have been able to use information about easily computed edge and label bounds on maximin goodness to provide a "sense of direction" to an existing relaxation algorithm. This reduces the effort involved in propagating label bounds to their related edge bounds. We have used similar information to prune post-relaxation search spaces.

References

[Dechter and Pearl, 1987] Dechter, R. and Pearl, J. Network-based heuristics for constraint satisfaction problems. *Artif. Intell.*, 34(1):1–38, December 1987.

[Dyson, 1980] Dyson, R.G. Maximin programming, fuzzy linear programming and multi-criteria decision making. *J. Op. Res. Soc.*, 31(3):263–267, March 1980.

[Haralick *et al.*, 1978] Haralick, R.M., Davis, L.S., Rosenfeld, A. and Milgram, D.L. Reduction operations for constraint satisfaction. *Inform. Sci.*, 14(3):199–219, June 1978.

[Rosenfeld *et al.*, 1976] Rosenfeld, A., Hummel, R.A. and Zucker, S.W. Scene labeling by relaxation operations. *IEEE Trans. Syst. Man & Cybern.*, 6(6):420–433, June 1976.

[Waltz, 1975] Waltz, D. Understanding line drawings of scenes with shadows. In P.H. Winston (ed.), *The Psychology of Computer Vision*. New York: McGraw-Hill, 1975.

[Yager, 1988] Yager, R.R. Some extensions of constraint propagation of label sets. Technical Report MII-801, Machine Intelligence Institute, Iona College, New Rochelle, NY 10801 USA, 1988.

FROM LOCAL TO GLOBAL CONSISTENCY*

Rina Dechter

Computer Science Department
Technion - Israel Institute of Technology
Haifa 32000, Israel
e-mail: dechter@techsel.bitnet

Abstract

In reasoning tasks involving the maintenance of consistent databases (so called "constraint networks") it is customary to enforce local consistency conditions in order to simplify the construction of a globally coherent model of the data. In this paper we present a relationship between the sizes of the variables' domains and the level of local consistency sufficient to ensure global consistency. The results are presented for binary constraint networks first, and then are generalized to higher order constraints. An interesting result emerging from this relationship is that any relation on bi-valued variables which is not representable by a network of binary constraints cannot be represented by networks with hidden variables, regardless of the number of hidden variables allowed.

1. Introduction

One of the major forces shaping resource-bounded reasoning stem from the requirement of local computation, namely, from the need to consider only a few data items at any inference step and to avoid the decision where to store intermediate results. All realistic models of human reasoning invoke this notion of locality in one form or another. For example, spreading activation in conceptual memories is grounded in the notion that activity spreads locally, among conceptually neighboring entities, but does not leap toward remotely designated addresses.

In reasoning tasks involving the maintenance of consistent set of information items (so called "constraints") the principle of locality has been embodied in techniques that enforce local consistency among groups of related variables. The rationale being that such local consistency will simplify the task of construction a globally coherent model of the data. For example, Truth Maintenance Systems [Doyle 1979], often sacrifice completeness by limiting the inferential steps to those involving constraint-propagation [McAllester 1980]. Such local techniques share the computational advantages mentioned earlier; there are only a few data items participating in each inference step, these items bear meaningful conceptual relationships to one another, partial results are stored exactly where they will be useful, computational steps can be performed in any order, and there is no need to remember which part of the knowledge has been processed and which part has not.

So far the conditions under which local consistency would entail global consistency involved topological properties of the network which represent the interaction among data items (so called "constraint networks") [Freuder 1982, Dechter 1987, Dechter 1989a]. In this paper we present a relationship between the sizes of the variables' domains in the network and the level of local consistency required for ensuring global consistency. The results are presented for binary constraint networks first, and then are generalized to higher arity constraints.

2. Definitions and Preliminaries

A **constraint-network (CN),** involves a set of n variables $X_1, \ldots, X_n$, each represented by its domain values, $D_1,...,D_n$, and a set of constraints. A constraint $C_i(X_{i_1},...,X_{i_j})$ is a subset of the cartesian product $D_{i_1} \times \cdots \times D_{i_j}$ that specifies which values of the variables are compatible with each other. A solution is an assignment of values to all the variables which satisfy all the constraints, and the most common task associated with these problems is to find one or all solutions. A constraint is usually represented by the set of all tuples which are not forbidden by it. A constraint network is associated with a relation ρ containing all its solutions. A **binary CN** is one in which all the constraints are binary, i.e., they involve only pairs of variables. An r-ary CN involves constraints with arity r or

*This work was supported in part by the National Science Foundation, Grant #IRI-8821444 and by the Air Force Office of Scientific Research, Grant #AFOSR-88-0177 while the author was visiting the cognitive systems lab at UCLA.

less.

A partial instantiation of variables $(X_1 = x_1, ..., X_i = x_i)$ is **locally consistent** if it satisfies all the constraints which are defined on subsets of $\{X_1, ..., X_i\}$. Any subset of variables X determines a subnetwork R_X that contains all the constraints defined on subsets of variables of X. A partial instantiation $(X_1 = x_1, ..., X_i = x_i)$ is **consistent** w.r.t. a containing subnetwork, R_X, if it is locally consistent and if it can be consistently extended to a full solution of R_X. A partial instantiation $(X_1 = x_1, ..., X_i = x_i)$ is **globally consistent** if it is consistent w.r.t. to the overall constraint network.

A network of constraints is said to be i-consistent [Freuder 1982] if every subnetwork of size $i-1$ which is locally consistent can be consistently extended by any i^{th} variable. Namely, if $(x_1, ..., x_{i-1})$ is locally consistent then for each variable X_i, there exists $x_i \in D_i$ such that $(x_1, ..., x_{i-1}, x_i)$ is locally consistent. A network is **strong i−consistent** if it is j-consistent for every $j = 1,2, ...i$.

When a network is not i-consistent, consistency-enforcing algorithms can be used to get it to the required consistency level [Mackworth 1984]. These algorithms process all subsets of i variables, and on each they record an $i-1$-ary constraint that ensures its consistency. The general complexity of an i-consistency enforcing algorithm is $\Omega((nk)^i)$ and $O((nk)^{2i})$.

A binary constraint network is the **minimal network** [Montanari 1974] if each locally consistent pair of values is globally consistent. A network of constraints is **globally consistent** if every locally-consistent subtuple is globally consistent. Clearly a network is i-consistent for all i iff it is globally consistent.

The notion of a globally consistent network is a generalization of Montanari's decomposability property which was defined for the minimal network. Global consistency means that the constraint network is completely explicit, namely, the set of all partial instantiations that are consistent with any subnetwork is exactly the projection of the overall relation on the corresponding subset of variables. An important property of globally-consistent networks is that they can be solved by a greedy search algorithm which is guaranteed to generate a solution without any dead-ends.

3. Local Consistency in Binary Networks

In this and in the following sections we present a relationship between the number of values in the domain of each variable and the level of local consistency that can ensure global-consistency. Theorem 1 presents the relationship for binary networks while Theorem 4 extends it to arbitrary networks. For simplicity and w.l.o.g. we assume that all variables have the same size, k, for their domains.

Theorem 1: A k-valued binary constraint network which is strongly $k+1$-consistent is also $k+i+1$-consistent for any $i \geq 0$.

Proof: Let $\bar{x} = (x_1, x_2, ..., x_{k+i})$ be a locally consistent subtuple of the subset of variables $\{X_1, X_2, ..., X_{k+i}\}$, and let X_{k+i+1} be an additional variable. We need to show that, given a k-valued network which is strongly $k+1$-consistent, there is a value x_{k+i+1} of X_{k+i+1} that is consistent with $\bar{x}$. Since all constraints are binary it means that the value x_{k+i+1} has to be consistent with each value of the set $\{x_1, x_2, ..., x_{k+i}\}$, separately. Variable X_{k+i+1} has k values $\{1,2, ..., k\}$ and we accordingly define k sets $A_1, ..., A_k$ as follows. A_j is the set of all values of $\bar{x}$ that are consistent with the value j of X_{k+i+1}. Since the problem is in particular 2-consistent, all $\bar{x}$'s values must appear in the set $A_1 \cup A_2, ..., \cup A_k$. We claim that there must be at least one set, say A_1, that spans all values $x_1, ..., x_{k+i}$. If this is not the case each set A_j must have a value x'_j that does not appear in it. From this it can be concluded that the tuple $\bar{x'} = (x'_1, x'_2, ..., x'_k)$ is not consistent with any one value of X_{k+i+1} which leads to a contradiction since we assumed strong $k+1$-consistency. Note that if the $\{x'_j\}$'s are not distinct we have a subtuple whose arity is less than k however since **strong** $k+1$-consistency was required the argument still holds.

Assume, now, w.l.o.g. that A_1 spans all values, therefore each value x_j, $j \in \{1, ..., k+i\}$ is consistent with the value "1" of X_{i+k+1} and we therefore found a value $(1 \in D_{k+i+1})$ which is consistent with $\bar{x}$. □

We may be tempted to conclude from Theorem 1 that any k-valued network can be solved polynomially by enforcing strong $k+1$-consistency. However enforcing $k+1$-consistency may require the addition of non-binary constraints for which the theorem doesn't apply. We can therefore conclude something weaker.

Corollary 1: A k-valued binary network, whose all induced constraints, (generated by enforcing strong-$k+1$-consistency) are decomposable to binary constraints, can be solved in $O((nk)^{2(k+1)})$ steps.

Proof: One can enforce strong $k+1$-consistency in $O((nk)^{2(k+1)})$ steps [Mackworth 1984]. Then, each recorded constraint is decomposed to its minimal network (by projecting the constraint on all subsets of two variables). Since we assumed that such a binary network represents each induced constraint precisely, the intersection of all the binary constraints result in a binary network of constraints which is strong $k+1$ consistent. Therefore, from Theorem 1 we can conclude that the resulting network is globally consistent. Since a globally consistent network can be solved in a backtrack-free manner, the cost of generating a solution is the cost of verifying that a partial instantiation of variables satisfies all the constraints. In binary networks this can be accomplished by $O(n^2)$ constraint checks. □

A unique special case exists when all variables are bi-valued, namely, $k = 2$. This is the only case when the network is inherently binary. According to Theorem 1, bi-valued networks require strong 3-consistency (also called path-consistency) in order to be globally consistent and 3-consistency can be achieved using binary constraints only. This leads to the following corollary:

Corollary 2: A strong 3-consistent bi-valued binary network is minimal.

It follows that:
Theorem 2: A bi-valued binary constraint network can be solved in $O(n^3)$ steps.

Proof: In this case enforcing strong 3-consistency takes $O(n^3)$, and since the resulting network is globally consistent, solution generation is $O(n^2)$. □

Corollary 2 has a surprising implication regarding the expressiveness of bi-valued binary constraint networks. It is quite obvious that most relations defined over bi-valued variables cannot be represented by a network of binary relations. Given a bi-valued relation ρ, we can generate the minimal network of ρ [Montanari 1974] by taking the projection of ρ on each pair of variables (projection is defined ahead). The set of all solutions of this minimal network always contains ρ and it is the best binary network approximation to ρ. Clearly in this network each locally consistent pair of values is globally consistent. When the minimal network represents ρ exactly we say that ρ is **binary-network-decomposable.** In most cases, however, a relation ρ will not be binary-network decomposable. The question is whether by allowing auxiliary bi-valued variables we can enhance the expressiveness of binary networks, namely, whether ρ can be a projection of some other relation which is binary-network decomposable. It turns out that if a bi-valued relation is not network-decomposable, adding any number of additional bi-valued variables will not remedy the situation. To formalize our claim we need the following definitions.

Let $rel(R)$ denote the relation associated with a network R, (i.e., $rel(R)$ is the set of all solutions to R). Let ρ be an n-ary bi-valued relation over a set of variables $X = \{X_1,...,X_n\}$. Relation ρ is **h-network-decomposable** if there exist h additional bi-valued variables $Y = \{Y_1,...,Y_h\}$ for which there is a binary network $R(X,Y)$ on $X \cup Y$ s.t. $\rho = \Pi_X rel(R(X,Y))$. $\Pi_U(\rho)$ denotes the projection[1] of relation ρ on subset of variables U. The additional variables needed for decomposition are called **hidden variables.**

Theorem 3: A bi-valued relation that is not network-decomposable is also not h-network-decomposable, for any h.

Proof: Assume the contrary, that ρ is a bi-valued relation which is not network-decomposable over variables $X = \{X_1,...,X_n\}$ and let $Y = \{Y_1,...,Y_h\}$ be a set of hidden variables such that there is a relation ρ′ over $X \cup Y$ satisfying $\rho = \Pi_X \rho'$ and ρ′ is network-decomposable. Since ρ′ is network-decomposable its minimal network, $M(\rho')$, must be a binary network decomposition of ρ′ and since it is minimal it is known to be strong 3-consistent [Montanari 1974]. The subnetwork, M_X, which is M restricted to the subset of variables, X, is a constraint subnetwork. According to Theorem 1, since M is bi-valued strong 3-consistent binary network, any tuple which is consistent in any subnetwork is globally consistent. In particular, partial solutions of M_X are globally consistent and therefore $rel(M_X)$ is the projection of the set of all solutions ($rel(M)$) on the set of variables X. Namely:

$$rel(M_X) = \Pi_X rel(M). \quad (1)$$

Since $\rho' = rel(M)$, we have

$$rel(M_X) = \Pi_X \rho'. \quad (2)$$

Since we assumed $\rho = \Pi_X \rho'$ we get:

$$\rho = rel(M_X) \quad (3)$$

M_X is, therefore, an exact network decomposition of ρ which contradicts our supposition. □

(1) The projection of a relation ρ on a subset of variables $U = U_1,...,U_l$ is given by $\Pi_U(\rho) = \{x_u = (x_{u_1},...,x_{u_l}) \mid \exists\ \bar{x} \in \rho,\ \bar{x}$ is an extension of $x_u\}$.

This result may severely limit the expressive power of some connectionist models, e.g., the Hopfield model, [Hopfield 1982] where each unit is assumed to have two states. Note however that in the Hopfield model the constraints are not necessarily binary and, therefore, further analysis of networks with higher order constraints is required.

4. Local Consistency in General Networks

In trying to generalize Theorem 1 to networks having constraints of arbitrary arity we follow the proof of Theorem 1 and make the needed modifications.

Theorem 4: A k-valued r-ary constraint network that is strong $k(r-1)+1$-consistent is also $k(r-1)+i+1$-consistent for any $i > 0$.

Proof: Let $\bar{x} = (x_1, x_2, ..., x_{k(r-1)+i})$ be a locally consistent subtuple over variables $\{X_1, X_2, ..., X_{k(r-1)+i}\}$, and let $X_{k(r-1)+i+1}$ be an additional variable. We need to show that, given a k-valued strongly $k(r-1)+1$-consistent network, there is a value $x_{k(r-1)+i+1}$ of $X_{k(r-1)+i+1}$ that is consistent with $\bar{x}$. Since the network has constraints of arity r or less, it means that such $x_{k(r-1)+i+1}$ has to be independently consistent with each subset of $r-1$ values of the set $\{x_1, x_2, ..., x_{k(r-1)+i}\}$, and the consistency of such sets is verified via the relevant constraints, having arity r or less, which are defined on the variable $X_{k(r-1)+i+1}$ and on the corresponding subset of $r-1$ variables from $\{X_1, X_2, ..., X_{k(r-1)+i}\}$. In other words, all the constraints having arity r or less, that involve variable $X_{k(r-1)+i+1}$ have to be verified in checking the consistency of any extension. Each such constraint can be uniquely determined by a subset of $r-1$ or less variables from the set $\{X_1, ..., X_{k(r-1)+i}\}$ on which it is defined.

Variable $X_{k(r-1)+i+1}$ has k values $\{1, 2, ..., k\}$ and we accordingly define k sets $A_1, ..., A_k$ as follows. A_j is the set of all subtuples of $\bar{x}$ of size less or equal to $(r-1)$ that are locally consistent with the value j of $X_{k(r-1)+i+1}$. Note that each such subtuple must be locally consistent and since the network is strongly $k(r-1)+1$-consistent, and in particular strong r-consistent, any such partial tuple must have at least one matching value in $X_{k(r-1)+i+1}$. Therefore all partial tuples of $\bar{x}$, having length $(r-1)$ or less must appear in the set $A_1 \cup A_2, \ldots, \cup A_k$. The participation of a tuple t in a set A_j means that all the constraints involving both the tuple's variables and variable $X_{k(r-1)+i+1}$ have to be satisfied. Let S denotes the set of all constraints that needs to be verified, the subset of constraints involved in the consistency verification of a tuple t with the value j of $X_{k(r-1)+i+1}$ by $S_t(j)$, and let $S(j)$ denotes all the constraints that are relevant to all tuples in A_j, namely $S(j) = \bigcup_{t \in A_j} S_t(j)$. Clearly $S = \bigcup_j S(j)$.

We claim that there must be at least one set, say A_1, that requires that **all** the constraints in S will be verified, namely that $S = S(1)$. If this is not the case each set A_j must have a constraint in S that is not verified. Let $\{X_{1j}, ..., X_{ij}\}$, $i \le r-1$, denote a constraint that is not verified in A_j. (Remember that constraints in S are identified by the subset of variables on which they are defined.) This means that the tuple $\bar{x}_j = (x_{1j}, ..., x_{ij})$ is not in A_j. From this it can be concluded that the tuple which results from concatenating all these excluded subtuples, $\bar{x}_1 \bar{x}_2 \cdots \bar{x}_k$ is not consistent with any of the values of $X_{k(r-1)+i+1}$. However, the length of this tuple is less or equal to $k(r-1)$ and since we assumed strong $k(r-1)+1$ consistency it must be consistent with at least one value, thus yielding a contradiction. Assume, now, w.l.o.g. that $S = S(1)$, therefore all partial tuples of $\bar{x}$ of size $r-1$ or less are consistent with the value "1" of $X_{k(r-1)+i+1}$ and we therefore found a value $(1 \in D_{k(r-1)+i+1})$ which is consistent with $\bar{x}$.

$\square$

Theorem 4 supplies us with the ability to generate complexity conjectures for families of constraint networks. For constraint networks having maximum number of values, k, and maximum constraint arity, r, we may try to prove that strong $k(r-1)+1$-consistency can be achieved using constraints with arity that does not exceed r. If this is proved Theorem 4 ensures that the resulting network is globally consistent and in such a case we have a polynomial problem with degree $2(k(r-1)+1)$.

An interesting special case arise in the context of k-colorability of a given graph. Any such decision problem corresponds to a binary constraint network that is **already k–consistent**. From Theorem 4 it is implied that if only we could extend the consistency level from k to $k+1$ the problem could be solved polynomially.

We can divide k-valued binary constraint problems having n variables into $\log_k n+1$ classes. The 0-class contains those problems that can be made $k+1$-consistent with only binary constraints, class 1 contains problems that can be made k^2 consistent using constraints of arity k or less, and in general, the i^{th} class contains problems that can be made (k^{i+1}) consistent via constraints of arity k^i or less. From Theorem 4 it follows that problems in class i can be solved in $O((nk)^{2k^{i+1}})$. Determining the class membership of a problem is still exponential, while verifying that the problem is in a given bounded class is polynomial.

Theorem 5: The membership of a problem in class i can be determined in $O((nk)^{2k^{i+1}})$.

Proof: First, we run the problem through a strong k^{i+1} consistency algorithm which takes $O((nk)^{2k^{i+1}}))$. The resulting problem may have constraints of arity k^i or less. For each such constraint we want to determine whether or not it is expressible with constraint of arity k^{i-1}. Let C be one such constraint. We then generate a network, $R_C^{(i)}$ of k^{i-1} -ary constraints by projecting C on each subset of k^{i-1} variables, an operation which is bounded by $O(k^{k^i})$, (i.e., the cardinality of the constraint C). We then have to solve the resulting network $R_C^{(i)}$ in order to check if it has the same solution set as the original relation C. Since this is a constraint network problem having k^i variables and k values, solving it is bounded by $O(k^{k^i})$. The number of constraints, like C, that we may need to process that way is $O(n^{k^i})$ and we get, therefore, that the overall complexity is $O((nk)^{2k^{i+1}})$. □

In the next subsection we show that by requiring only directional consistency we are able to bound the complexity to a squareroot of the above expression.

5. Directional Consistency

The notion of directional consistency was introduced in [Dechter 1987] as a mean of weakening local consistency demand while still ensuring global consistency along a given ordering. The advantage of this weaker consistency property is that it can be enforced much more cheaply and at the same time it ensures backtrack-free search using the same ordering.

Definition: A network of constraints is said to be directional i-consistent, w.r.t. an ordering $d = X_1,...,X_n$ if every subnetwork of size $i-1$ which is locally consistent can be consistently extended by any i^{th} variable which succeeds all the subnetwork's variables in the ordering d. A network of constraints is **directional globally consistent** w.r.t. ordering d, if it is directional-i-consistent for every i.

Theorem 6: A k-valued r-ary constraint network that is directional strong $k(r-1)+1$-consistent w.r.t. d is also directional $(k(r-1)+i+1)$-consistent for any i.

Proof: Follows immediately from the proof of Theorem 4. □

For completeness sake we present an algorithm, **adaptive(level)** that enforce directional strong $i+1$-consistency when $level=i$, which was presented at [Dechter 1989b]. Let *level* be a parameter indicating the utmost cardinality of constraints which are recorded. Let d be an ordering $X_1, \ldots, X_n$, and let PARENTS(X_i) be the variables preceding X_i in the ordering d which are connected to it (originally these are all the variables that are explicitly constraining X_i).

```
adaptive(level, $X_1, \ldots, X_n$)
Begin
1. for i=n to 1 by -1 do
2. Compute PARENTS($X_i$)
3. perform new-record( $level, X_i$, PARENTS($X_i$))
4. for $level \geq 2$, connect all elements in PARENTS($X_i$)
(if they are not yet connected)
End
```

Procedure new-record(*level*, var, set) records only constraints of size less or equal to *level* from subsets of *set* and is defined as follows:

```
new-record(level, var, set)
Begin
1. if $level \leq |set|$ then
2.   for every subset $S$ in $set$, s.t $|S| = level$ do
3.     record-constraint($var$,$S$)
4.   end
5. else do record-constraint(var,$set$)
end
```

The procedure **record-constraint(V,SET)** generates and records those tuples of variables in SET that are consistent with at least one value of V. Clearly, the algorithm enforces a directional strong $level+1$ consistency. The complexity of adaptive(*level*) is both time and space dominated by the procedure new-record(*level*) which is $O(\binom{n}{level} \cdot (k^{level+1})) = O((nk)^{level+1})$.

Given an ordering d of an r-ary k-valued constraint network, a problem is in **directional class** i if directional strong $k^{i+1}(r-1)+1$ consistency can be enforced using constraints of arity $k^i(r-1)$ or less. As we saw, the cost of enforcing this level of directional consistency by *adaptive* $(k^{i+1}(r-1))$ is $O((nk)^{k^{i+1}(r-1)+1})$. We get, therefore, that the bound on directional consistency is a square-root of the bound on full consistency and we can conclude that:

Theorem 7: Given an ordering of the variables, d, the membership of a problem in directional class i can be determined in $O((nk)^{k^{i+1}(r-1)+1})$ steps. □

6. Examples

Tasks of reasoning with time and space provide many examples of constraint networks, having special local consistency properties. Allen's algebra [Allen 1983], for example, defines thirteen possible relations between time intervals and provides a transitivity table for their propagation. This algebra can be described as a traditional constraint network where the variables are the relationships between two intervals, each having 13 values, and the transitivity table defines ternary constraints on triplets of variables. Having $k = 13$ and $r = 3$, we can conclude from Theorem 4 that if we can enforce 27-strong-consistency without introducing higher than ternary constraints, we can generate a globally consistent network using a polynomial algorithm of degree 27. However, since Allen's algebra is known to be NP-complete it follows that this is not an achievable task.

A second example comes from simplifying temporal constraints into relationships between time points. Vilain and Kautz [Vilain 1986] suggested that if we consider three temporal relations between any two time points $\{<,=,>\}$, and define the transitive relationship induced by such relations we get a simpler network that can be made globally consistent using a 3-consistent algorithm. This claim was later corrected by van Beek and Cohen [van_Beek 1989], showing that 4-consistency is necessary for global consistency.

This PA^* algebra (as termed by van Beek et al.) can be described as a traditional constraint network, where the variables are the relationships between two points and the transitivity table defines ternary constraints. This yields a constraint network with $k = 3$, $r = 3$. Theorem 4 suggests that if such a network is 7-strong-consistent it is globally consistent. To make this observation useful one has to show that it is feasible to enforce 7-consistency with ternary constraints. This is indeed the case since van Beek and Cohen [van_Beek 1989] have shown that even 4-consistency is sufficient for global consistency, and this is achievable via ternary constraints. Although the result of van Beek et al. is much stronger, our result is a direct bi-product of a general principle.

If we further simplify the point algebra and consider only two labels between time points $\{<,>\}$ (one may argue that equality never really happens), we get a ternary 2-valued constraint network. According to Theorem 4, strong 4-consistency is sufficient to ensure global consistency and, being a subset of PA^*, this level of consistency can indeed be achieved with ternary constraints only.

A different family of constraint networks arises in the domain of scene labeling. Huffman [Huffman 1971] and Clowes [Clowes 1971] developed a basic labeling scheme for blocks world picture graphs. Given a basic labeling set: + (convex), - (concave), -> (occluding, object on arrowhead side), and a standard set of simplifying assumptions on scene content, the physically realizable junction labeling are just those shown in Figure 1.

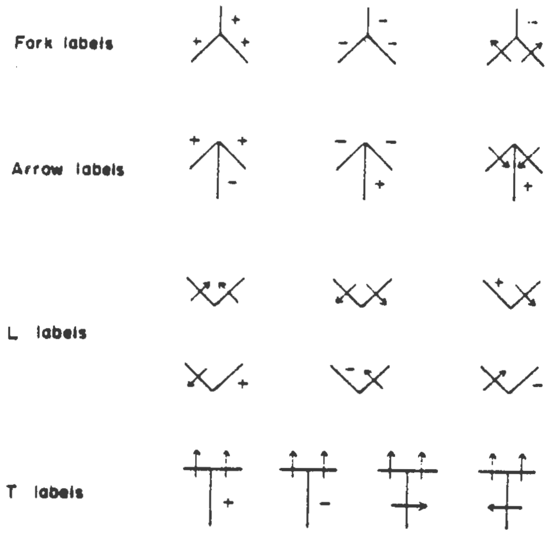


Figure 1: Junction Labels

Freuder [Freuder 1980] provided algorithms for labeling this restricted set while Waltz [Waltz 1975] explored a richer label set.

A network composed of junctions in Figure 1 can be viewed in two ways: Each line can be viewed as a variable having three values while the constraints are binary or ternary depending on whether two or three lines intersect. In this view Theorem 4 states that if a given network is already 7-strong-consistent it is already globally consistent. The second view treats each junction as a variable, each variable has 3 or 4 values (the number of possible labeling combination of a junction) and the constraints are binary. Theorem 1 states that if the problem is 4-consistent it is globally consistent. The second view, thus, provided a much weaker consistency demand for guaranteeing global-consistency. However, it is yet unclear whether this level of consistency is achievable without increasing the arity of the constraints.

7. Conclusions

A globally consistent network permits the construction of a consistent solution in linear time. We showed that the amount of local consistency required for achieving such global consistency is dependent on the product of two parameters: the number of values in each variable and the constraint-arity. The complexity of achieving the required local consistency is exponential in this product.

A surprising implication emerging from the above relationship is that for the special case when the constraints are bi-valued and binary ($k=r=2$), 3-consistency ensures global consistency in all cases. As a consequence we showed that if a bi-valued relations is not representable by a binary constraint network it cannot be helped by any number of hidden variables.

Acknowledgement

I would like to thank Itay Meiri for his assistance in formulating Theorem 4.

References

[Allen 1983] Allen, J.F, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, Vol. 26, No. 11, 1983, pp. 832-843.

[Clowes 1971] Clowes, M. B., "On seeing things," *Artificial Intelligence*, Vol. 2, 1971, pp. 79-116.

[Dechter 1987] Dechter, R. and J. Pearl, "Network-based Heuristics for Constraint-Catisfaction Problems," *Artificial Intelligence*, Vol. 34, No. 1, 1987, pp. 1-38.

[Dechter 1989a] Dechter, R. and J. Pearl, "Tree Clustering for Constraint Networks," in *Artificial Intelligence*, 1989, pp. 353-366.

[Dechter 1989b] Dechter, R. and I. Meiri, "Experimental evaluation of preprocessing techniques in constraint satisfaction problems," in *Proceedings of the 11th Intl. Conf. on AI (Ijcai-89*, Detroit, Michigan: 1989.

[Doyle 1979] Doyle, J., "A Truth Maintenance System," *Artificial Intelligence*, Vol. 12, 1979, pp. 231-272.

[Freuder 1980] Freuder, E.C., "On the knowledge required to label a picture graph," *Artificiall Intelligence Journal*, Vol. 15, No. 1, 1980, pp. 1-17.

[Freuder 1982] Freuder, E.C., "A Sufficient Condition for Backtrack-Free Search.," *Journal of the ACM*, Vol. 29, No. 1, 1982, pp. 24-32.

[Hopfield 1982] Hopfield, J.J., "Neural networks and physical systems with emergent collective computational abilities," *Natl. A cademy of Sciende, USA*, Vol. 79, 1982, pp. 2554-2558.

[Huffman 1971] Huffman, D. A., "Impossible objects as nonsense sentences," *B Meltzer and D. Michie Eds., Machine Intelligence*, Vol. 6, 1971, pp. 195-234.

[Mackworth 1984] Mackworth, A.K. and E.C. Freuder, "The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems," *Artificial Intelligence* , Vol. 25, No. 1, 1984.

[McAllester 1980] McAllester, D. A., "An outlook on truth-maintenance," MIT, Boston, Massachusetts, Tech. Rep. AI Memo No. 551, 1980.

[Montanari 1974] Montanari, U., "Networks of Constraints: Fundamental Properties and Applications to Picture Processing," *Information Science*, Vol. 7, 1974, pp. 95-132.

[van_Beek 1989] van_Beek, P. and R. Cohen, "Approximation algorithms for temporal reasoning," in *Proceedings Ijcai-89*, Detroit, Michigen: 1989.

[Vilain 1986] Vilain, M. and H. Kautz, "Constraint propagation algorithms for temporal reasoning," in *Proceedings AAAI-86*, , Phila, PA.: 1986, pp. 377-382.

[Waltz 1975] Waltz, D., "Understanding Line Drawings of Scenes with Shadows," in *The Psychology of Computer Vision*, P. H. Winston, Ed. New York, NY: McGraw-Hill Book Company, 1975.

Reconstructing Polyhedral Scenes from Single Two-Dimensional Images: The Orthogonality Hypothesis

Jan A. Mulder

Department of Mathematics, Statistics,
and Computing Science
Dalhousie University
Halifax N.S.
Canada B3H 3J5

Robert J. MacG. Dawson

Department of Mathematics,
and Computing Science
St. Mary's University
Halifax N.S.
Canada B3H 3C3

Abstract

This paper proposes the orthogonality hypothesis: provided certain two-dimensional constraints are satisfied, some trihedral junctions in a polyhedral scene are assumed to represent orthogonal corners. Using this hypothesis one can analytically infer the three-dimensional coordinates of the vertices representing orthogonal corners. A reconstruction algorithm is presented which uses the orthogonality hypothesis to initiate reconstruction at a few vertices and then propagates the reconstruction to other vertices in the scene. A formal analysis of this algorithm indicates some of its strengths and limitations.

1. Introduction

The process of projecting a three-dimensional (3-D) scene onto a two-dimensional (2-D) image is mathematically relatively straightforward and well known. The reverse process of how to recover the 3-D properties of a scene given a 2-D image, is not as well understood. Psychologists have traditionally emphasized the role of perceptual organization in the 3-D reconstruction process. However, elements of a computational theory of the role of perceptual organization in 3-D reconstruction have started to emerge only recently (Witkin and Tenenbaum, 1983, 1986; Perkins, 1983). Although the overall basis of perceptual organization is not well understood, there appears to be some consensus that one of its purposes is to detect stable image groupings, which reflect actual structure in the scene rather than accidental properties. Stability in image groupings means that such groupings remain invariant over a variety of viewpoints. Groupings have been proposed based on different image relations such as proximity, parallelism, collinearity (Lowe, 1987), and skewed symmetry (Kanade, 1981).

In this paper we propose a heuristic principle suitable for reconstructing many types of scene: the *orthogonality hypothesis*. This recognizes that, while we may not have a global model of the objects in the scene, we may be able to assume as a *local* assumption that some vertices will be orthogonal. In addition, we show that this hypothesis can be used to reconstruct the 3-D coordinates of vertices and orientation of most edges in a polyhedral scene consisting of a variety of objects without using any specific knowledge about the objects involved. Finally, we formally analyze some of the strengths and limitations of this approach.

Orthogonality assumptions are known to facilitate the analysis of 3-D scenes. This has been demonstrated in work using gradient space representation (Mackworth, 1976), and in work in shape-from-contour (e.g. Barnard 1983; Horaud 1987, 1989). The main difference between the work presented here and previous work is an explicit exploration of orthogonality based on local assumptions only. In addition, this paper contains a formal analysis of some of the strengths and limitations of the orthogonality hypothesis when it is used in the reconstruction of polyhedral scenes. The method used is analytic, and uses orthographic projection only.

2. The Orthogonality Hypothesis

If we compare Fig. 1 with Fig. 2 then it is easily observed, that all corners of Fig. 1 could be orthogonal, whereas no such interpretation is allowable in Fig. 2. Perkins, in a study titled "Cubic Corners" (Perkins, 1968) discovered that in a polyhedral scene three-line junctions can represent orthogonal corners if and only if one of the following constraints is satisfied:

1. All three angles are obtuse.
2. Two angles are right.
3. Two angles are acute and the third is less than 270 degrees.

From here on we will refer to these three constraints as the *orthogonality constraints*. The mathematical correctness of Perkins's discovery is simple to prove. It is based on the observation that the angles of a projection of an orthogonal trihedral corner contain enough information to determine orientation up to sign.

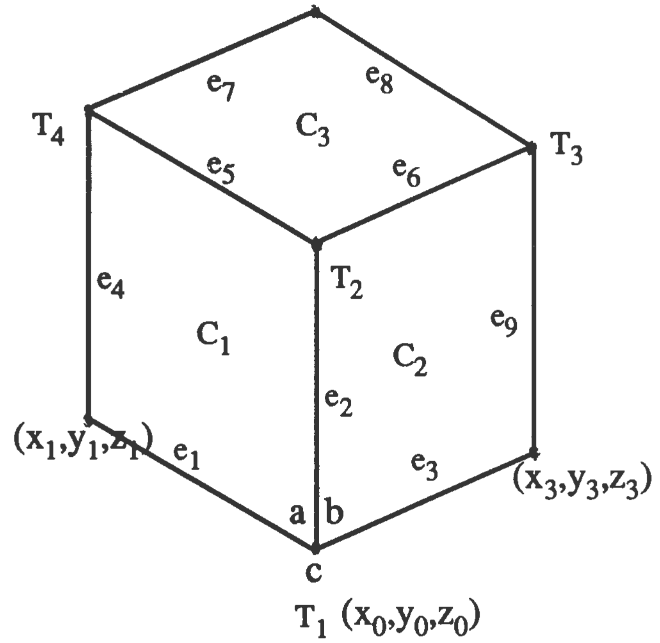


Figure 1

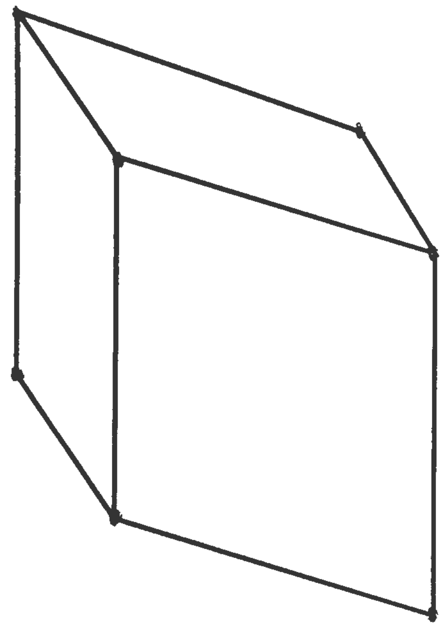


Figure 2

In the coordinate system which we will use, the projection of the corner in *XYZ* space is simply its orthogonal projection on the viewing *XY* plane. Perkins proves that the angles *a, b,* and *c* of the trihedral junction (see Fig. 1) represented by three vectors in the *XY* plane can be traced back to three vectors in *XYZ* space which are mutually orthogonal and project, respectively, to the vectors in the *XY* plane. People living in an industrialized society can distinguish orthogonal from non-orthogonal corners with a high degree of accuracy (Perkins, 1972). Native populations in Zimbabwe, on the other hand, have been found to score much lower on the rectangularity discrimination task (Perkins and Deregowski, 1982).

As a complement to Perkins's proof, Mulder and Dawson (1989) have shown that an orthogonality assumption can also be used to compute a viewpoint with respect to a trihedral corner. The only information needed are the angles *a, b,* and *c* of the trihedral junction. If the junction represents an orthogonal corner (taken to be the origin of *XYZ* space), then the coordinates of the viewpoint (up to scalar multiple and some changes of sign) are determined to be:

$$\sqrt{\cot b \cot c}, \qquad \sqrt{\cot a \cot c}, \qquad \sqrt{\cot a \cot b}$$

The orthogonality hypothesis can be defined as follows: *provided the orthogonality constraints are satisfied, trihedral junctions in a polyhedral scene are assumed to represent orthogonal corners. The constraints imposed on the scene by this assumption are propagated across the object of which the corner is a component.*

Figures 1 and 2 are examples of a rectangular and a non-rectangular object. All trihedral junctions in Fig. 1 satisfy the orthogonality constraints, whereas none of the junctions in Fig. 2 do. However, it is also possible for a trihedral junction to satisfy the orthogonality constraints, while it is not orthogonal in space. Fig. 3 is an example of such a situation. Both T_1 and T_2 satisfy the orthogonality constraints. Yet, only one of them can represent an orthogonal corner. The problem is caused by the fact that different corners in a single object should be consistent in viewpoint (Lowe, 1987). In Fig. 1 this is the case for all trihedral junctions, but not so in Fig. 3. T_1 and T_2 cannot be orthogonal from the same viewpoint. A consistent interpretation can therefore be arrived at only if one of the corners is accepted as non-orthogonal.

The orthogonality hypothesis is powerful. It helps us determine the 3-D coordinates of many vertices in the scene without having to rely on models of the objects in the scene. This scene recovery process takes place in an analytical manner and it uses orthographic projections only. In the next section we present an algorithm which uses orthogonality of a particular junction as a working hypothesis and which uses this hypothesis to propagate the 3-D position of vertices and orientation of edges constrained by the junction over the scene.

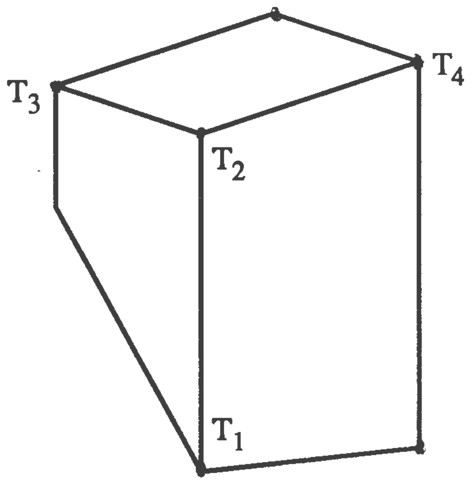


Figure 3

3. Propagating Orthogonality Constraints over the Scene

We will first describe the orthogonality constraint propagation process in an informal way before presenting a formal algorithm. Let us take Fig.1 as an example. For the purpose of explanation we assume that we are dealing with a perfectly segmented image; that is, we can compute all junctions *T* and 2-D closures *C* in the image. (A 2-D closure is a cycle of edges which is not internally bridged.) Each 2-D closure represents a surface in 3-D space. In 2-D space we use an *XY* coordinate frame with its origin in the lower left corner of the image.

In 3-D space we use the same coordinate system, but we add a z-coordinate perpendicular to the *XY* (image) plane. The advantage of such a coordinate system is that we can compute the relative z-coordinates of vertices in terms of the viewpoint computed with respect to a trihedral junction assumed to be orthogonal (Mulder and Dawson, 1989). In addition, in scenes with more than one object we can align objects by a simple translation of z-coordinates.

First, create a queue of all trihedral junctions which satisfy the orthogonality constraints. We exclude T-junctions from this process, because such junctions may represent edges belonging to different objects. For Fig. 1 the queue consists of the junctions: $(T_1\ T_2\ T_3\ T_4)$.

The first step is to delete T_1 from the queue and to reconstruct it by hypothesis, as an orthogonal corner. We assume the coordinates of T_1 to be $(x_0, y_0, 0)$ with x_0 and y_0 representing the image coordinates of T_1. The z-coordinate of the endpoints of e_1 can then be expressed as a function of x_0, y_0, x_1, y_1 and the angles a, b, c of T_1. For instance, z_1 can be computed from the following expression:

$$z_1 = \pm\sqrt{\frac{\cot a \cot c}{1 - \cot a \cot c}\quad ((x_1 - x_0)^2 + (y_1 - y_0)^2)}$$

z_2 and z_3 can be computed in a similar way. A positive value of z_1 assumes that the viewpoint is outside the object. This assumption is generally preferable to an "inside" viewpoint.

The edges e_1, e_2, e_3 jointly determine the 3-D orientation of the closures C_1 and C_2, represented in 3-D space by the surfaces S_1 and S_2. Knowing the orientation of S_1 and the end point coordinates of e_1 allows us to determine the coordinates of e_4 and e_5. In the same manner we can compute the end point coordinates of e_6 and e_9 in S_2. Next, the coordinates of the vertices in C_3 are inferred. Since e_5 and e_6 are now known, we have determined the orientation of S_3. This enables us to compute the location of the last missing edges, e_7 and e_8.

Since S_3 is directly constrained by S_1 and S_2, we amalgamate the three surfaces into a single cluster. In this paper, we define an object as a cluster of surfaces. An object, however, does not necessarily correspond to a solid object with an interior space.

The 3-D reconstruction of Fig. 1 could have taken place in more than one way. The analysis could also have been based on T_2, T_3, or T_4 which would have resulted in a similar viewpoint and a compatible set of coordinates for the vertices.

We will now discuss the reconstruction algorithm. We will then proceed to show that this algorithm can interpret scenes with more than one object (such as Fig. 4) correctly.

Definition: Two edges are 3-D-connected if they are connected by means of a junction or L-vertex and neither edge forms the stem of a T-junction.

The Reconstruction Algorithm RA*:

```
1. Determine all 2-D closures $C$ ($C_1$ ......... $C_k$) in the image.
   ($C_0$ is the outer-region and is not part of $C$).
2. Create a queue of all trihedral junctions (with the exclusion
   of T-junctions) which satisfy the orthogonality con-
   straints. Call this queue $T$ with elements $T_1$ ..... $T_n$. The
   order of vertices is arbitrary.

3. While $T$ not empty do:
   3.1 Mark each $C_i$ in $C$ uncompleted ($1 <= i <= k$).
   3.2 Delete the 1st element $T_i$ ($1 <= i <= n$) from $T$. Com-
       pute a viewpoint $V_i$ (for $T_i$) which constitutes a hy-
       pothesis $H_i$ for the local 3-D situation. We attempt to
       create clusters of surfaces based on $H_i$.
```

```
3.3 Compute the 3-D coordinates of the vertices connected with $T_i$.
3.4 Create a queue $UC$ consisting of all uncompleted $C_i$
    ($1 <= i <= k$) which contain 2 or more 3-D-connected edges with known 3-D location.
3.5 Create a cluster $CL_i$ for surfaces constrained by $H_i$.
    $CL_i \leftarrow \{ \}$.
3.6 WHILE $UC$ not empty do:
    3.6.1 Delete the first element $c$ from $UC$.
          Create a surface $s$ which represents $c$ in 3-D space.
          Propagate $(c,s)$.
          Mark $c$ completed.
          Associate $H_i$ with $s$.
          Append $s$ to $CL_i$.
    3.6.2 $UC \leftarrow$ union of $UC$ and all uncompleted $C_i$
          which contain two or more 3-D-connected edges with known 3-D location.
3.7 For each $T_j$ ($1 <= j <= n$, $j \neq i$) with all three of its edges constrained by $H_i$ do:
    If $|V_j| = |V_i|$ then delete $T_j$ from $T$.
```

Every combination of non-overlapping clusters covering all surfaces constitutes a valid scene interpretation.

```
Procedure Propagate $(c,s)$
1. Among the set of edges surrounding $c$ find 2 connecting edges with known 3-D location such that the connecting junction is a trihedral junction which is not a T-junction. Call this junction $tr$.
2. Assume $c$ is surrounded by a sequence of $m$ edges ($E_1$ ...... $E_m$) such that $E_1$ and $E_m$ are edges connecting in $tr$.
3. Compute the orientation of $s$ based on the end point coordinates of $E_1$ and $E_m$.
4. FOR $k = 2$ TO $m$ DO
     IF $E_k$ and $E_{k-1}$ are 3-D-connected
        THEN compute the 3-D location of $E_k$ based on the end point coordinates of $E_{k-1}$ and the orientation of $s$.
5. FOR $k = m-1$ DOWNTO 2 DO
     IF $E_{k+1}$ and $E_k$ are 3-D-connected AND the 3-D location of $E_k$ is unknown
        THEN compute the 3-D location of $E_k$ based on the end point coordinates of $E_{k+1}$ and the orientation of $s$.
```

The reconstruction algorithm is conservative. Vertex coordinates are not propagated across the stem of a T-junction, because T-junctions may indicate occlusion by another object. The intent of cluster formation in RA* is to join surfaces which are likely to belong to a single object. The computation of the 3-D location of all edges surrounding a surface implies complete visibility of the surface. Step 3.7 in RA* is an efficiency measure. It prevents duplication of interpretations.

Fig. 4 is an example of a scene with two objects partially occluding each other. The reconstruction algorithm starts with junction T_1 and determines the coordinates of the vertices surrounding C_1 and C_2. At this point C_3 becomes part of UC and the coordinates of its vertices are computed as well. The cluster CL_1 now consists of the surfaces S_1, S_2, and S_3, and grows no further. As a side effect, the junctions T_6, T_7, and T_8 are removed from T (step 3.7).

Next, we delete T_2 from T. Now the coordinates of

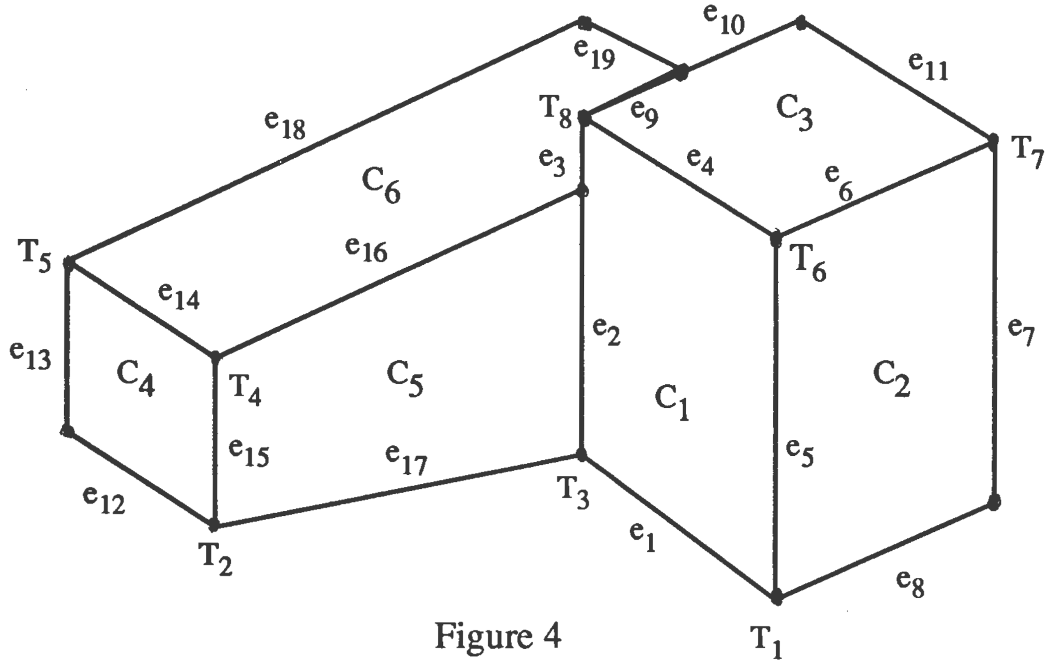


Figure 4

the vertices surrounding C_4 and C_5 are determined. This causes C_6 to be added to UC and the coordinates of its vertices are computed. Note, that because of the 3-D-connectedness rule the edges e_3 and e_9, although part of C_6, are not associated with S_6, thereby emphasizing S_6's partial visibility. The end point coordinates of these edges are computed as part of S_1 and S_3 respectively.

When more than one object is involved, some interesting issues arise about the precedence of principles of perceptual organization. The viewpoints from which T_2 and T_4 are orthogonal do not match. Two different surface clusters (CL_2 and CL_4) are therefore created, each based on a different viewpoint assumption. Two different global interpretations arise from this split. In the first one (CL_1 CL_4) S_3 and S_6 have the same orientation, in the second interpretation (CL_1 CL_2) they do not. The parallelism principle stipulates that lines parallel in the image are also parallel in the scene. In the (CL_1 CL_4) interpretation orthogonality and parallelism cooperate. In the (CL_1 $CL2$) interpretation orthogonality conflicts with parallelism.

Alignment of objects also becomes an issue when more than one object is involved. For every orthogonality hypothesis we assume a zero value for the z-coordinate of the corresponding junction. If we assume (from the proximity principle) that two objects touch each other, then the z-coordinates in one of the objects must be realigned. If the objects in Fig. 4 touch in e_2 then we must translate the z-coordinates of one of the objects such that z-coordinates are the same in T_3. This realignment will reveal the spatial nature of two T-junctions; e_{16} turns out to be connected with e_2 and e_3, whereas e_{19} does not connect with e_9 and e_{10}. Thus, RA* is capable of resolving connection/occlusion issues.

4. A formal analysis of RA*.

Despite RA*'s apparent ability to reconstruct polyhedral scenes with a variety of objects, the algorithm has limitations as well. To mention a few:

1. A sufficient supply of "orthogonal" trihedral junctions (i.e. trihedral junctions which are not T-junctions, and which satisfy the orthogonality constraints) must be available. More precisely, every object in the scene must contain at least one "orthogonal" trihedral junction.
2. Higher order junctions (i.e. junctions with more than 3 lines) can block the propagation process. For example, RA* may fail in pyramid shaped objects with more than three visible surfaces. On the other hand, if a higher order junction lies on the boundary between two or more objects, then RA* may survive.
3. T-junctions can block reconstruction. The procedure "propagate" is successful as long as there is at most one edge forming the stem of a T-junction on the boundary of the surface explored. A second T-junction stem will prevent RA* from completing the propagation. Usually, such a situation implies that part of the surface is occluded by another object (e.g. S_6 in Fig. 4).

The strength and limitations of RA* can be proven by means of the following theorem.

Theorem :

The reconstruction algorithm will recover the coordinates of all visible edges of a single object, if the following conditions are met:

1. The edges of the object are straight and its surfaces are planar.
2. Edges meet in L-vertices and trihedral junctions only.
3. At most one edge on the boundary of each surface forms the stem of a T-junction.
4. At least one trihedral junction is not a T-junction, and it satisfies the orthogonality constraints.

Note that these conditions exclude pyramid shaped objects with more than two visible surfaces. In addition, if the object has more than two visible surfaces, then each surface must be adjacent to at least two other surfaces which are also adjacent.

Proof:

The proof is in three parts. First, we prove that at least two adjacent surfaces can be completed (i.e. the coordinates of all visible edges surrounding the surface are computed). Next, we show that at any time, before RA* terminates, there is an uncompleted surface that has at least two 3-D-connected edges with known coordinates. Finally, we prove that RA* terminates.

1. Condition 4 stipulates that there is at least one trihedral junction *tr* which satisfies the orthogonality constraints. The presence of *tr* implies that at least two of the object's surfaces (say, S_1 and S_2) are visible. If we assume *tr* to be orthogonal, then we can compute the coordinates of the three vertices it is connected with. The edges connecting these vertices determine the orientation of both S_1 and S_2. Thus, we can compute the coordinates of all vertices of the adjacent surfaces S_1 and S_2. If the object has only two visible surfaces then we have thus computed the coordinates of all vertices.
2. If the object has more than two visible surfaces, then each surface is adjacent to at least two others, which are also adjacent. Thus, there must be a surface (S_3) adjacent to both S_1 and S_2. Since all of the vertices of S_1 and S_2 are known, the end points of two 3-D-connected edges of S_3 must be known as well. This condition is sufficient to complete S_3.
3. RA* has two WHILE loops. In the outer loop the elements of T are removed one by one and are not replaced. With a finite number of elements in T, T must eventually get empty. At the start of the inner WHILE loop UC contains two closures each consisting of at least two 3-D-connected edges with known location. In principle, all closures of the object may be placed on the queue. However, once a closure is removed from UC, it is completed and will not be

returned. *UC* must therefore eventually get empty as well. Thus, both WHILE loops are guaranteed to terminate. As a result, RA* must terminate.

One limitation of this theorem is that it only applies to objects when seen in isolation. How does RA* behave, when the scene consists of a variety of objects, which are touching and/or partially occluding each other? For this situation, we can somewhat relax the conditions without sacrificing any of RA*'s power. Subject to condition 4 being satisfied for each object in the scene we can relax condition 2 by permitting higher order junctions provided that they mark the boundary between two or more objects. This does not cause any difficulty for the propagation of vertex coordinates, because propagation can be blocked only by the stem of a T-junction. As an example, Fig. 5 is successfully interpreted by RA*. The interpretation results in 2 clusters: CL_1 containing S_1, S_2, and S_3, and CL_2 containing S_4, S_5, and S_6. Note that e_4 will obtain 2 different sets of coordinates depending on whether it forms the boundary of S_1 or S_6.

Fig. 5 also raises some interesting issues regarding precedence of principles. If we assume the two objects to be wedges, then the orthogonality hypothesis conflicts with the parallelism principle. Assuming both principles, on the other hand, implies concavity for one of the objects. Proximity is also an issue. If we assume the two wedges to connect at the lower end of e_4, then the two objects are not connected at the higher end.

Fig. 6 illustrates not only the strengths, but also some limitations of RA*. If we omit edge e_8 from the image, then the presence of one trihedral junction satisfying the orthogonality constraints suffices to determine the complete 3-D layout of the object. However, if we add e_8 to Fig. 6, then we create two objects. For the object on the left, T_1 will still compute the coordinates of all vertices. However, the object on the right (consisting of C_3 and C_4) has no valid trihedral junction, since T-junctions are discarded. As a result, nothing can be done in the second object.

The main strength of the orthogonality hypothesis, however, remains its emphasis on local models whose constraints are propagated across the scene. A reconstruction of the scene is therefore possible, even if the objects in the scene are unfamiliar. Thus, we can avoid one of the chicken and egg problems in perception, namely that we cannot make 3-D sense of an image unless we make assumptions about the objects in the scene first.

5. Possible extensions.

Given a projection of a polyhedral object, one can have many potential reconstructions under the orthogonality hypothesis. In ambiguous situations, RA* will simply list all possible interpretations. People, on the other hand, tend to show preference for particular interpretations. It is possible, to modify RA*'s behavior to reflect this phenomenon. For one thing, we can take further advantage of the orthogonality hypothesis. If two or more trihedral junctions, via global reconstruction, suggest the same viewpoint, then clusters based on this viewpoint should be preferred over others. We may make this approach quantitative by assigning an *a priori* probability of $p > 0.5$ that any given junction satisfying the orthogonality constraints is orthogonal. The computed probability of an interpretation will then be greatest when it con-

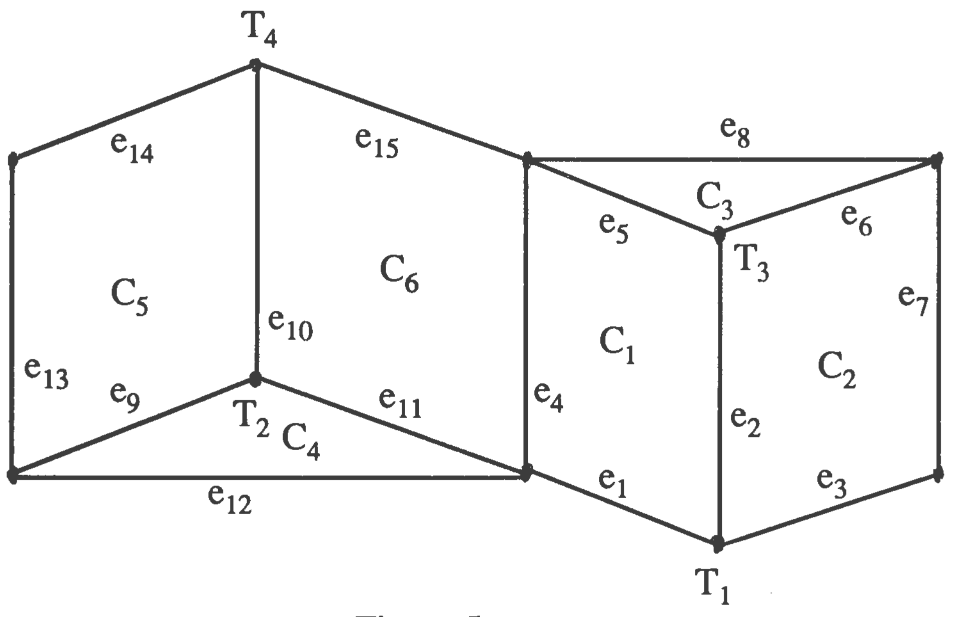


Figure 5

tains most orthogonal junctions.

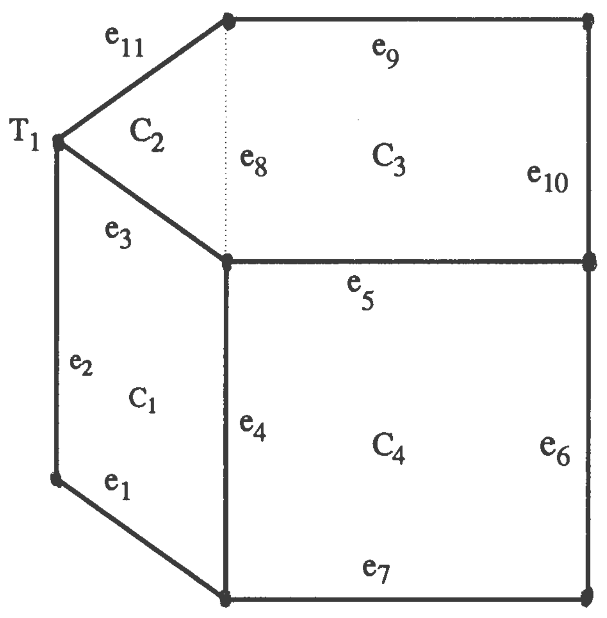


Figure 6

This approach can be further generalized by including other shapes for trihedral junctions, such as the 90:90:60 junction of a right triangular prism, the 90:90:120 junction of a right hexagonal prism, or the 60:60:60 junction of a regular tetrahedron. We may also use families of trihedral junctions: for instance, those with 1 or 2 right angles, or those with equal face angles.

Finally, RA* can be extended to cope with imperfect segmentation. Lowe (1987) has suggested a method that calculates the probability for two line segments with adjacent endpoints to have arisen by accident of viewpoint. This method can be extended to compute a certainty value for junctions in an imperfectly segmented image. Combined with *a priori* probabilities for corner models we can thus extend RA* to compute a probability for each interpretation and choose the interpretation with the highest probability.

6. Conclusion

In this paper, we have proposed the orthogonality hypothesis: provided certain constraints are satisfied some trihedral junctions in a polyhedral scene are assumed to represent orthogonal corners. Using an analytical method and orthographic projection, we can then infer the 3-D coordinates of the vertices connected with this corner. A reconstruction algorithm was presented which propagates vertex coordinates across the scene. The conditions under which this algorithm can recover the complete structure of the scene were analyzed. The orthogonality hypothesis allows for a reconstruction of a polyhedral scene using local models only.

7. Acknowledgements

This research has been supported by grant no OGP 0948 from the Natural Sciences and Engineering Research Council of Canada.

8. References

[1] T. Barnard, "Interpreting Perspective Images", *Artificial Intelligence,* 21, 435-462, 1983.

[2] R. Horaud, "New Methods for Matching 3-D Objects with Single Perspective Views", *IEEE Trans. On Pattern* Analysis and Machine Intelligence (PAMI), vol. 9,*401-412,*1987.

[3] R. Horaud, B. Conio, O. Leboulleux, B. Lacolle, "An Analytic Solution for the Perspective 4-Point Problem", IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 500-507, 1989.

[4] T. Kanade, "Recovery of the Three-Dimensional Shape of an Object from a Single View", *Artificial Intelligence,* 17, 409-460, 1981.

[5] D. Lowe, "Three-Dimensional Object Recognition from Single Two-Dimensional Images", *Artificial Intelligence,*31, 355-395, 1987.

[6] A.K. Mackworth, "Model-driven Interpretation in Intelligent Vision Systems", *Perception,* vol. 5, 49-370, 1976.

[7] J.A. Mulder, R.MacG. Dawson, "From Two-Dimensional Images to Three-Dimensional Models:The Orthogonality Principle for Trihedral Junctions", Technical Report 1989-CS5, Department of Mathematics, Statistics, and Computing Science, Dalhousie University,1989.

[8] D.N. Perkins, "Cubic Corners", M.I.T. Research Laboratory of Electronics, Quarterly Progress Report 89, 1968, pp. 207-214.

[9] D.N. Perkins, "Visual Discrimination betweenRectangular and Non-Rectangular Parallelopipeds", *Perception and Psychophysics,* 1972, 12, 396-400.

[10] D.N. Perkins, "Why The Human Perceiver Is A Bad Machine", in *Human and Machine Vision,* J. Beck, B. Hope, A. Rosenfeld, (eds.), Academic Press, NY, 1983, pp. 341-364.

[11] D.N. Perkins, J.B. Deregowski, "A Cross-cultural Comparison of the Use of a Gestalt Perceptual Strategy", *Perception,* vol. 11, no 3, 279-286, 1982.

[12] A.P. Witkin, J.M. Tenenbaum, "On the Role of Structure in Vision", in *Human and Machine Vision,* J. Beck, B. Hope, A. Rosenfeld (eds.), Academic Press,NY, 1983, pp. 481-544.

[13] A.P. Witkin, J.M. Tenenbaum, "On Perceptual Organization", in *From Pixels To Predicates,* A.P. Pentland (ed.), Ablex Publishing Company, Norwood, NJ, 1986, pp. 149 - 169.

Visual Structure Inference with Uncertainty

Paul R. Cooper
Institute for the Learning Sciences and
Department of Electrical Engineering and Computer Science
Northwestern University
1890 Maple Avenue
Evanston, IL 60201
cooper@ils.nwu.edu

Abstract

This paper addresses the problem of inferring the structure of a composed object given an evidential image-derivable description. The structure inference problem is posed as a labelling problem expressed in a Markov Random Field network. Evidence about high-level structural hypotheses is represented as likelihoods at the labels, and prior knowledge is represented as weights or clique parameters in the MRF. A network inference algorithm is then used to combine the evidence and prior knowledge to yield a segmented description of a Tinkertoy scene. Implementation experiments involving the traditionally difficult problems of occlusion and accidental alignment are given to demonstrate the effectiveness of the framework.

1 Introduction

Noise and the projection of a 3D world into two dimensions mean that image data can provide only uncertain information for use in higher level visual processing, such as recognition. Recognition therefore requires inference and decision making.

This paper describes a parallel network that infers the structure of a composed object from an uncertain description. In particular, a massively parallel representation with uncertainty is developed for Tinkertoy objects. A network inference mechanism is then used to combine image-derivable evidence with prior knowledge to yield a segmented description of a Tinkertoy scene.

Inferring the true physical structure of an object in the world is a necessary part of the process of recognizing it. In the typical approach, decisions (usually thresholds) are made during the building of an object description from image data, yielding a single symbolic object description that is hoped to be essentially correct.

In contrast, the focus in this work is reflecting the evidential or uncertain nature of the input data even in high-level object representations for structurally composed objects. Object structure hypotheses and the evidential support for the hypotheses are encoded in a parallel labelling framework. The labelling framework itself is represented as a Markov Random Field, which provides a principled probabilistic interpretation of the evidence and inference process. The Markov Random Field representation also allows the expression of appropriate prior knowledge in a convenient way. A discrete object description is then inferred from the evidence and prior knowledge.

2 Background

2.1 Recognition from Structure and Structure Inference

The recognition of Tinkertoy objects is the goal of The Tinkertoy Project. Previous work[Cooper and Swain, 1989; Swain and Cooper, 1988; Cooper, 1988; Cooper and Hollbach, 1987] has addressed issues in the parallel recognition of objects from structure, assuming that discrete essentially error-free descriptions of composed objects can be obtained from images. Recognition from structure addresses the role of parts and the spatial relations between them in the recognition process[Pentland, 1987; Biederman, 1985; Hoffman and Richards, 1986]. Because their identity is defined primarily by the spatial relationships between simple parts, Tinkertoys provide a convenient domain task for examining recognition from structure.

When perfect information assumptions are *not* made, developing a principled solution to the problem of recognition from structure is considerably more difficult. The solution of at least two related sub-problems is required. The first problem is determining the *identity* of the object in the scene, with respect to a model base of objects already known. This sub-problem must be solved even when absolutely perfect geometric descriptions of the object in the scene are available. The second sub-problem, present when only uncertain evidence is available, is inferring the *physical description of the object in the world* from the available evidence and relevant prior knowledge. It is this second sub-problem, structure inference, that is the topic of this paper.

(In other work, I argue that these are coupled problems, and show how both problems can be solved together simultaneously in a unified way[Cooper, 1989; Cooper, 1990]. Others have addressed the recognition of structured objects by developing heuristic inexact matching algorithms that compensate for the possibility of an erroneous object description[Shapiro and Haralick, 1981; Eshera and Fu, 1986]).

Inferring the true structure of the physical world subsumes both the problems of segmentation and reconstruction. Evidential approaches to these problems have been proposed before[Feldman and Yakimovsky, 1974; Chou, 1988; Sher, 1987], but usually with a much lower-level visual representation.

2.2 Markov Random Fields in Vision

Markov Random Fields (MRFs) have been used as the basis of an evidential approach to many computer vision tasks in recent years[Geman and Geman, 1984; Marroquin, September 1985; Cross and Jain, 1983; Chou, 1988]. While most previous vision work has used MRFs that are essentially rectangular arrays, the theory of Markov Random Fields applies to arbitrarily structured graphs like the one described later. Some of the relevant aspects of MRF theory and its application to labelling problems are now very briefly reviewed [Kindermann and Snell, 1980].

Consider a set $\mathbf{X}$ of discrete-valued random variables X. Associate with the random variables an undirected graph G defined as a set S of sites (or vertices) and a neighborhood system (or set of edges) E. The random variables of the field are indexed by the graph vertices as X_s. Variables are neighbors in the MRF when the associated vertices are adjacent in the graph. In the formulation of a labelling problem as an MRF, the variables in the labelling problem are the random variables of the MRF.

The value ω_s of a random variable may be any member l_i of the state space set L. Because of the application of the field to the labelling problem, the event elements of the set L will be called labels. An assignment of values to all the variables in the field is called a configuration, and is denoted ω.

We are interested in the probability distributions P over the random field $\mathbf{X}$. Markov Random Fields have a locality property:

$$\begin{aligned} P(X_s = \omega_s | X_r = \omega_r, r \in S, r \neq s) &= \\ P(X_s = \omega_s | X_r = \omega_r, r \in N_s) & \qquad (1) \end{aligned}$$

that says roughly that the state of site is dependent only upon the state of its neighbors (N_s). MRFs can also be characterized in terms of an energy function U with a Gibb's distribution:

$$P(\omega) = \frac{e^{-U(\omega)/T}}{Z} \qquad (2)$$

where T is the temperature, and Z is a normalizing constant.

If we are interested only in the prior distribution $P(\omega)$, the energy function U is defined as:

$$U(\omega) = \sum_{c \in C} V_c(\omega) \qquad (3)$$

where C is the set of cliques defined by the neighborhood graph G, and the V_c are the clique potentials.

Specifying the clique potentials V_c provides a convenient way to specify the global joint prior probability distribution P. The clique potentials can be conveniently viewed as weights in a connectionist network. They provide a mechanism to express soft constraints between labels at related variables. Unary clique potentials in effect express first order priors, while binary clique potentials express the constraints between pairs of variables in the field.

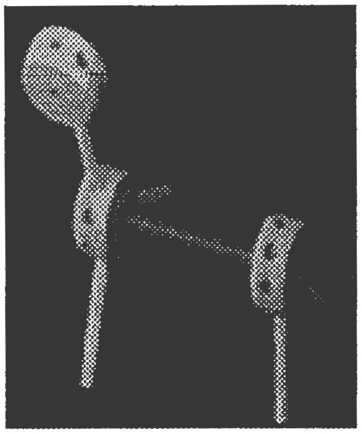


Figure 1: Real Tinkertoy Image

Suppose we are instead interested in the distribution $P(\omega|O)$ on the field after an observation O. An observation constitutes a combination of spatially distinct observations at each local site. The evidence from an observation at a site is denoted $P(O_s|\omega_s)$ and is called a likelihood. Assuming likelihoods are local and spatially distinct, it is reasonable to assume that they are conditionally independent. Then, with Bayes' Rule we can derive:

$$U(\omega|O) = \sum_{c \in C} V_c(\omega) - \sum_{s \in S} \log P(O_s|\omega_s) \qquad (4)$$

To summarize, the MRF represents a labelling problem. Evidence about the hypotheses is expressed as label likelihoods, and prior knowledge is expressed in terms of the clique potentials, generalized weights that express soft constraints between spatially related variables.

Inference on the MRF network can be framed in terms of the energy function. For example, the maximum a posteriori probability can be computed by finding the minimum of the non-convex energy function U. Needless to say, this is a non-trivial problem and not the focus of this work. In the experiments which follow, a deterministic approximation algorithm called HCF[Chou, 1988] is used to find a good minimum of the energy function. This minimum corresponds to a particular selection of labels for each variable.

3 Network Description

3.1 Network Structure

The crux of the problem is defining a network that can represent possible Tinkertoy objects, including uncertainty about their structure. As can be seen in Figure 1, even images of very simple Tinkertoy scenes contain significant uncertainty. Part parameters such as rod length are clearly difficult to determine reliably. Self-occlusion

Variables	Labels
slots	doesn't exist exists and empty exists and full
virtual rods	doesn't exist exists and has length L1 exists and has length L2 exists and has length L3

Table 1: Definition of Variables and Labels

and noise make the geometry of the junctions difficult to determine, and this might have a profound effect on the task of determining the identity of the object. In short, in a real image of even a very simple part-and-junction world, the existence, identity and parameterization of the parts might all be uncertain.

Many possible network designs are possible. The chief constraints on the design arise from trying to represent the relevant uncertainties while minimizing the complexity of the network. A more detailed and rigorous presentation of the network, including complexity analysis, is available in Cooper [1989].

The most basic element of the network design is the definition of the variables and labels. Adopting the unit/value principle[Barlow, 1972; Feldman and Ballard, 1982; Ballard, 1984] leads to a discrete parameter representation of Tinkertoy part-and-junction geometry with two types of variables: rods and slots [Cooper and Swain, 1989]. Slot variables represent the slots in Tinkertoy junction disks where rods can be connected, and provide a discrete representation of possible junction geometries. Uncertainty about rod length is represented by labels at the rod variables. Structural uncertainty is represented by whether or not slots and rods are connected. The possibility of a rod connecting any two slots is represented by the existence of a rod variable for every pair of slots, so-called "virtual rods". With this representation, most mutually exclusive hypotheses are competing labels at a variable (eg. different rod lengths). The competition between other mutually exclusive hypotheses (eg. virtual rods representing alternative connections to the same slot) is enforced by winner-take-all network topology[Feldman and Ballard, 1982].

The definitions of the variables and labels are summarized in Table 1. Note that the possibility that parts do not exist (they might be artifacts of signal noise, for example), is explicitly represented.

The key to any connectionist design is establishing connections which encode the essence of the computation. In the case of the MRF, the connections define the MRF graph, and represent adjacencies between related variables in the field. In this design, there are three kinds of connections. Slot/slot connections communicate consistency amongst the interpretations of the slot variables at a single disk junction. For example, all slots at a disk should be consistently labelled as "existing". Virtual rod/virtual rod connections consitute a winner-take-all network. Each virtual rod is connected to all the other incompatible hypotheses and discourages them. Finally, there are connections between related slots and virtual

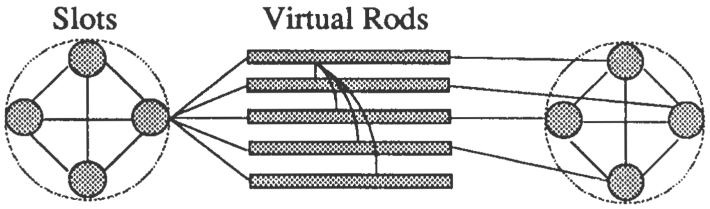


Figure 2: Fragment of MRF Graph. The shaded objects are the MRF sites (slots and rods). The solid lines represent edges in the MRF graph. The virtual rods show all the connections from one slot to slots on one other disk, as well as the dangling rod possibility. The set of virtual rods is a clique; only some of the connections are shown.

rods, so that communication between rod/slot pairs can enforce consistency. For example, a consistent rod/slot pair is one where the rod "exists" and is connected to a "exists and full" slot.

A fragment of the full MRF graph defined in this way is given in Figure 2. The irregular nature of the graph structure reflects the application of representing high-level structure, and differs greatly from traditional image-based array MRF applications.

3.2 Likelihoods: The Image-derivable Evidence

To complete the definition of the MRF, it is necessary to provide data for the field. The evidence and prior knowledge must be specified.

Variable/label events (eg. rod of length L1) were carefully chosen to represent spatially local events in the image. Thus, it should be possible to develop operators that gather likelihood evidence about the events from the image. Significant work exists that demonstrates the feasibility of building such probabilistically well-founded likelihood operators [Bolles, 1977; Sher, 1987], at least for lower-level events such as edge and line detection. However, providing well-justified evidence about higher-level events requires a fully developed probabilistic model of low and intermediate level vision, including evidence combination for composed events. The eventual generation of high-level evidence via such a theory was assumed for this work, and evidence was synthesized artificially. Generally the evidence was constructed from qualitative criteria, such as "very certain", "almost no evidence for this", etc. The experiments deliberately probed a wide range of input conditions, reflecting possible data that could arise from real images. Furthermore, the network behavior was extremely robust to variations in the exact values for the evidential input, reducing the significance of the fact that the data were synthetic.

3.3 Prior Knowledge

In any inference problem involving perception, there are only two sources of information: sensor evidence for this problem instance, and what was known before. In probabilistic frameworks including MRFs, previous knowledge is expressed as priors. The joint prior distribution on an entire MRF is expressed through the clique potentials, or

Clique		Potential
Slot	Slot	
doesn't exist	doesn't exist	consistent
doesn't exist	exists, full	inconsistent
doesn't exist	exists, empty	inconsistent
exists, empty	exists, empty	freq. dependent
exists, empty	exists, full	freq. dependent
exists, full	exists, full	freq. dependent
Vrod	Vrod	
doesn't exist	doesn't exist	consistent
doesn't exist	exists, $L1\|L2\|L3$	consistent
exists, $L1\|L2\|L3$	exists, $L1\|L2\|L3$	inconsistent
Slot	Vrod	
doesn't exist	doesn't exist	consistent
doesn't exist	exists, $L1\|L2\|L3$	inconsistent
exists, empty	doesn't exist	consistent
exists, empty	exists, $L1\|L2\|L3$	inconsistent
exists, full	doesn't exist	inconsistent
exists, full	exists, $L1\|L2\|L3$	good

Table 2: 2-Clique Potentials

network weights. Domain dependent Tinkertoy knowledge, both qualitative and quantitative, was represented by clique potentials in the Tinkertoy MRF. Although it should be possible to derive clique potentials directly from frequency measurements on problem instances in the world, clique potentials were synthesized on an ad hoc basis.

The first set of potentials represented constraints arising from the nature of Tinkertoys. For example, for a disk to exist in the world, all its slots must have the "exists" label. Clearly, we know this fact *a priori*. In another example, the "existing filled" hypothesis at a slot is consistent with the existence of an adjacent rod. A table summarizing a qualitative description of the potentials for cliques of size 2 is shown in Table 2. In practice, once the appropriate order of magnitude was established for the representation of such clique parameters, variations in the values of the potentials had no effect on network performance.

The network architecture also allows the expression of quantitative prior knowledge where it is appropriate. The frequency with which local properties occurred in past problem instances is representable as clique potentials. For example, first order statistics about the lengths of rods in previous problem instances were encoded as clique potentials at the length labels of the rod variables. Salient second and higher order features (such as junction geometry at a disk) were also represented in the network. In this way, statistics based on a domain of previous problem instances can influence perceptual inference in the current problem instance. One might think of the domain depedent prior knowledge as "smoothing" the current evidence to a solution during the inference process on the network.

4 Experiments

An implementation of the MRF network described above was built with the Rochester Connectionist Simulator[Goddard *et al.*, 1988].

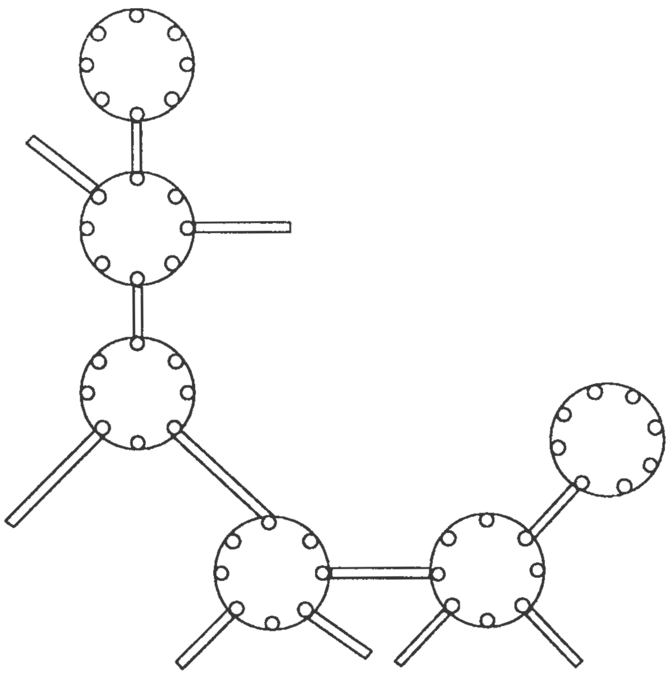


Figure 3: A Man and His Dog: schematic of image showing accidental alignment

4.1 Accidental Alignment: A Man and His Dog

The basic structure of the first experiment is given in Figure 3. The scene shows a 2D Tinkertoy scene with 2 objects, a man and his dog, accidentally aligned so that the hypothesis that there is only one object is reasonable. (Alternative interpretations that have been suggested include a caterpillar and a molecule with 6 atoms).

The experiment is principally designed to explore and demonstrate the power of the representational framework. The scene represents a non-trivial problem of representation and inference. The solution framework provides the richness necessary to represent the ambiguity arising from a degenerate alignment of objects, and provides an inference mechanism strong enough to overcome the ambiguity and make a decision.

Some interesting features of the experiment are as follows. First, it demonstrates a scene in which a segmentation ambiguity is present, and shows how the evidence in such a scene might occur and be represented. Local labelling ambiguity is present and simple to represent — for example, different rod lengths have different likelihoods. The experiment also contains non-trivial structural uncertainty — is it one object or two? Second, it demonstrates the way both priors and evidence combine to yield a decision. In this experiment, the evidence about the major segmentation decision is (by design) inconclusive. The priors must therefore provide the information necessary to achieve an interpretation. This represents one possible balance that can exist between the evidence and the priors. In some cases with ambiguous evidence, prior knowledge alone is inadequate, and interpretation mistakes are made. Thirdly, it demonstrates

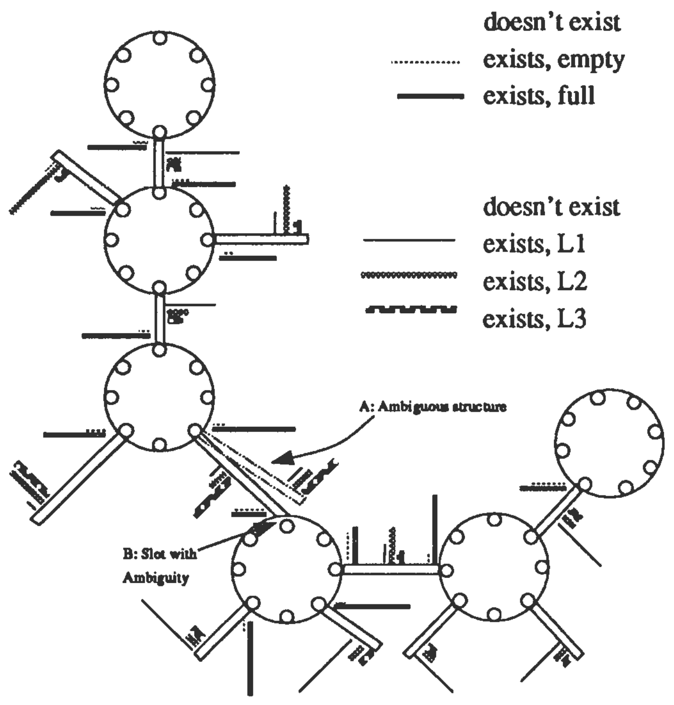


Figure 4: Input Evidence from the Image: bar graphs of label likelihoods

the power of the inference procedure in resolving such an ambiguous decision problem. The sequential trace of the inference process is particularly impressive in this regard, because it involves a local decision-reversal. In this case, the global energy of the later decision is better than the first decision.

The input evidence for the problem instance is presented graphically in Figure 4. In the figure, the likelihoods are shown for each label by bar graphs located near the spatial hypothesis they describe. The lines are scaled to represent likelihoods between zero and one.

The evidence surrounding the connection of the two objects (at point A in the figure) is very ambiguous. The 'connected' and 'disconnected' hypotheses both have very similar evidence. Note that the hypothetical likelihood generator was fairly confident about the length of the rod, just not about whether it connected the two slots or not. (Both hypotheses have about the same likelihoods at each of their labels). This is an example of how true structural ambiguity is represented in the net. Note also that the evidence at the slot hypothesis (B in the figure) is completely ambiguous. In effect, because of the accidental alignment, the likelihood generator would find evidence for the *full* label. On the other hand, a reasonable likelihood generator would probably have knowledge about slot-rod junctions, and would thus know that lack of perpendicularity at the junction is evidence for the *empty* label.

In Figures 5 through 9 the progress of the inference process on the experiment is shown. Figure 5 shows the first few label commitments that were made: these are the MRF sites with the least ambiguous evidence.

Of particular interest is Figure 6 when HCF has in-

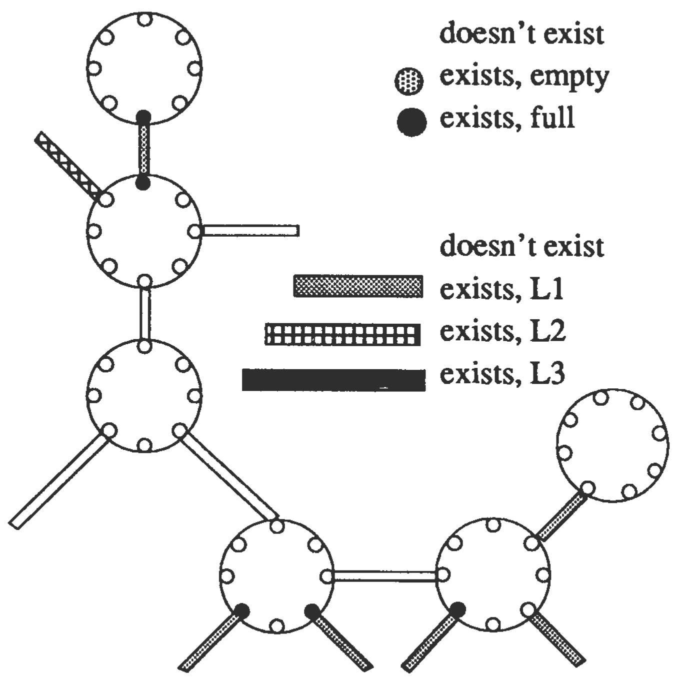


Figure 5: Segmentation Inference on MRF: First Few Committed States

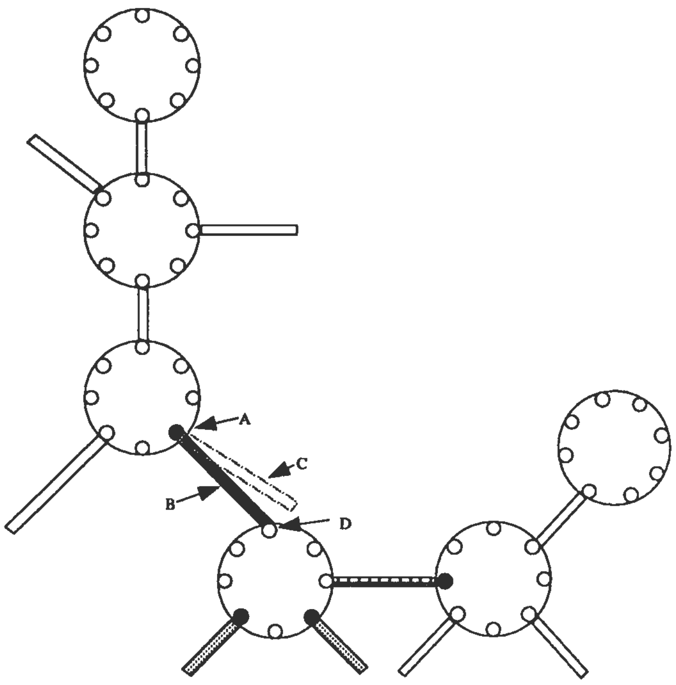


Figure 6: Segmentation Inference on MRF: Incorrrect vrod hypothesis "B" is chosen over correct hypothesis "C", providing excitation energy for the "full" label at "D"

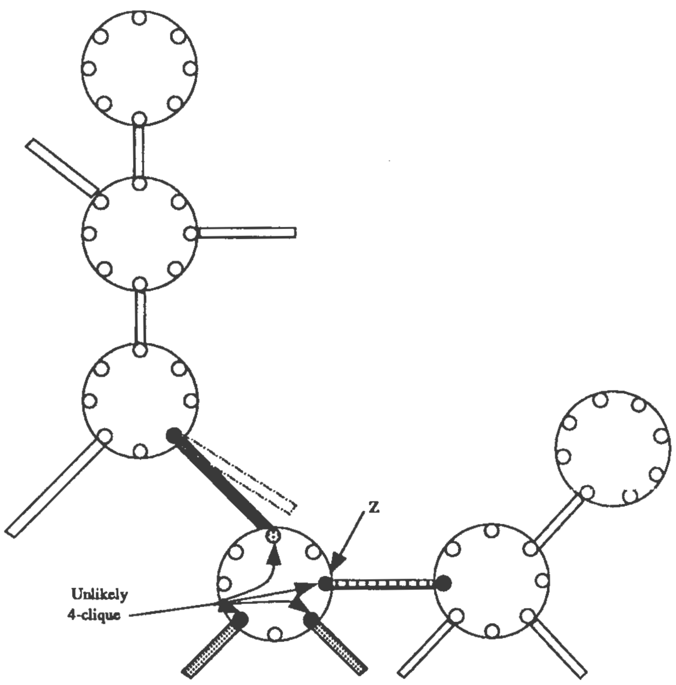


Figure 7: Segmentation Inference on MRF: "full" decision at "Z" makes inhibitory 4-clique relevant

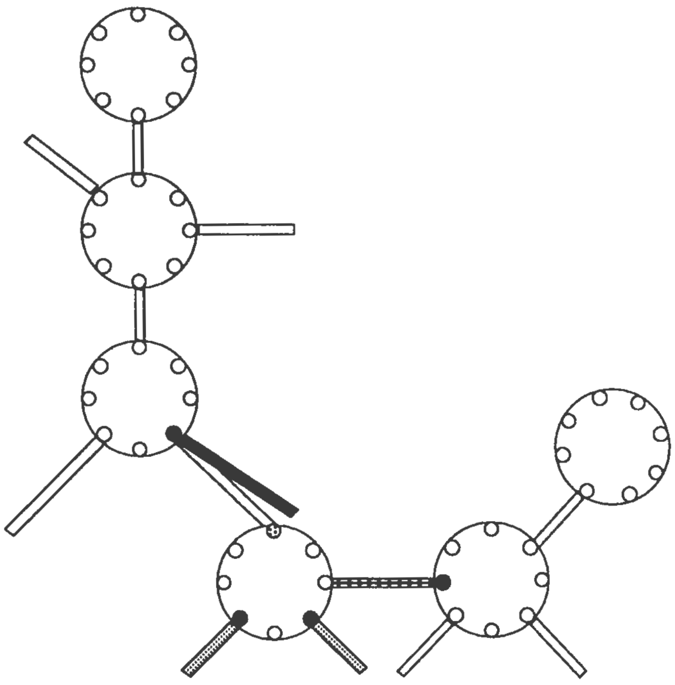


Figure 8: Segmentation Inference on MRF: inconsistent "empty" slot with connecting vrod causes change of decision

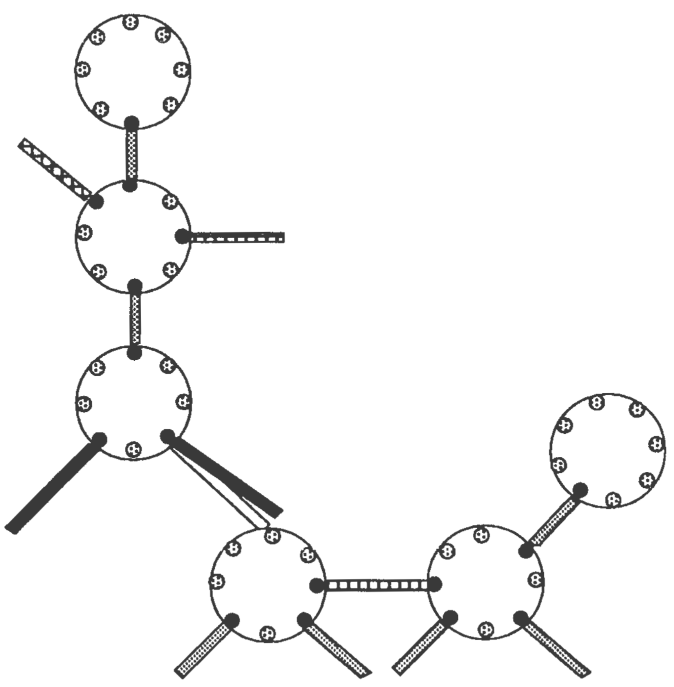


Figure 9: Final Segmentation

correctly labelled the scene as one object. Once the slot (the man's 'hip' joint, designated A in the figure) is labelled *full*, excitation energy exists in favor of some rod being attached. Then the 'connecting' virtual rod (B) is chosen because its local evidence is slightly stronger than that of the competing 'dangling' virtual rod (C). When the connecting virtual rod is labelled as existing, the associated unlabelled slot D gets energy encouraging the *full* label.

The slot D does not get labelled immediately. Instead, later, the slot designated Z in Figure 7 gets labelled as *full*. Once the slot Z is *full*, if slot *D* were full as well, this would commit 4 slots on that disk as *full*. But the set of priors used for the experiment states that this particular configuration of 4 rods at a disk is unlikely, as marked in Figure 7. This 4-clique prior inhibits the simultaneous labelling of all 4 of the slots as *full*. The inhibition energy is sufficient to commit the fourth slot, slot D, to the state *empty*, as shown in Figure 7. Of course, an empty slot is incompatible with the vrod B hypothesizing connection, so that vrod gets relabelled as *not existing*, and the alternative rod C is relabelled as *existing and L3*.

Other parts of the input evidence are comparatively ambiguous as well. The man's right leg, for example, has very uncertain evidence about the correct length. (An explanatory assumption for the evidence might be that that region of the image was noisy, so the likelihood generator had difficulty discriminating lengths). This uncertainty, purely local, is easily resolved by the network.

Eventually, the whole scene is correctly labelled, as shown in Figure 9.

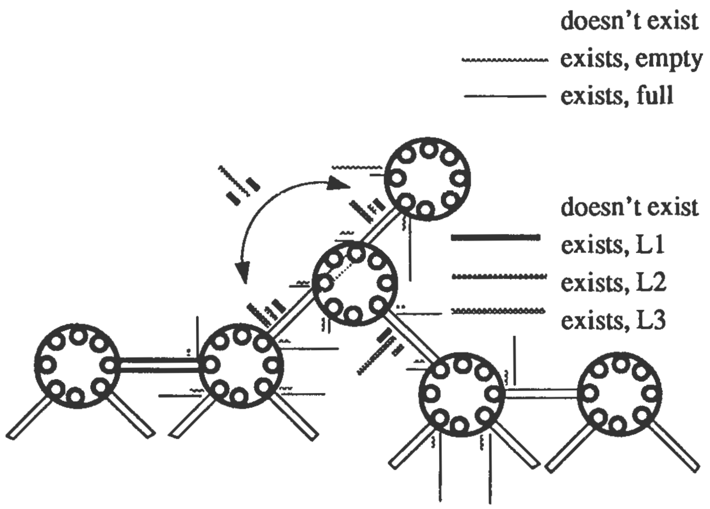


Figure 10: Dog Occluding Giraffe, Input Evidence: label likelihoods

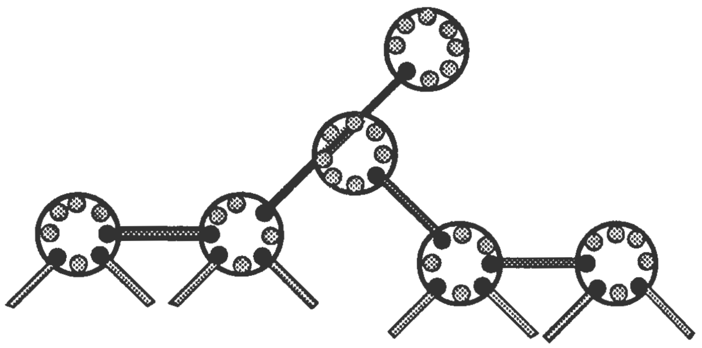


Figure 11: Dog and Giraffe: Final Segmentation Labelling

4.2 Dog Occluding Giraffe

A more representative experiment is now given. This experiment demonstrates the kind of uncertain information that could arise in a typical occlusion, and how it might be resolved. Compared to the extremely unlikely accidental alignment that was correctly interpreted in the last experiment, this occlusion is a much simpler problem for the system to handle. Relative to the capabilities of other vision systems, especially those not handling uncertainty, the ability to correctly segment an occluded scene is a non-trivial achievement.

The scene and the input evidence can be seen in Figure 10. Note the evidence about a variety of structural hypotheses surrounding the location of the occlusion. In order for there to be any possibility at all of a mistaken interpretation, it is once again necessary to contrive an accidental alignment of the occluded rod and two occluding slots. (Otherwise, the likelihood generators would not have ambiguous local information, and the correct global interpretation would follow simply from the evidence).

Let's examine the evidence relevant to the occlusion. At a high-level of analysis, there are two main possible hypotheses: the true hypothesis (dog occluding giraffe) with a single occluded rod connecting the giraffe's head to its shoulders, and the mistaken hypothesis that the giraffe and dog are awkwardly connected. The latter hypothesis requires two short rod's connecting the giraffe's head and shoulders to the dog, respectively. Note that the local evidence about the vrods representing the hypotheses is ambiguous; they have exactly the same likelihoods. Consider also the evidence about the slots. On the dog's head, the two slots aligned with the occluding rod show better evidence for *full* than *empty*. The local evidence is thus actually ranked in inverse order to the truth.

For this problem, making a segmentation decision based on the usual simple criteria will obviously not suffice. In particular, a threshold will yield the *wrong* answer, because of the incorrect ranking of the evidence at the slots.

As can be seen in Figure 11, the Tinkertoy MRF eventually achieves the right interpretation, correctly segmenting the two overlapping objects. In this problem, the statistically derived priors reflecting the frequency of junction-pattern occurrence play little role. Instead, constraints propagating to the competing rod hypotheses at the occlusion allow the correct interpretation. Consider the "giraffe neck" hypothesis versus its competitors. The evidence at the slot at both ends of the neck is strongly in favor of the slots being labelled *exists and full*. As a result, HCF commits those states early. At this point, the connecting "neck" virtual rod is receiving excitation energy from *both* slots. The competing (incorrect) hypotheses are each compatible at *one end only*. As a result, the "neck" hypothesis commits to existence, winning the vrod WTA competition, and forcing the competitors to turn off. In short, the correct global interpretation is more compatible with the evidence, and thus has a lower energy.

5 Conclusion

This paper has shown a high-level parallel representation for complex structurally composed objects that incorporates a principled treatment of uncertainty and domain dependence. Given the fundamentally evidential nature of image data, inference decisions about the world must be made at some point. Rather than depend upon heuristically selected thresholds, this framework demonstrates how uncertainty may be organized and maintained at a high level of representational abstraction. Inference decisions about object structure can then incorporate appropriately high-level prior knowledge. As a result, correct segmentation decisions can be computed even when the local evidence is ambiguous or favors the incorrect interpretation.

Experiments have been shown that demonstrate the power of combining evidence and prior knowledge at a high-level of abstraction. Visual inference decisions were computed that would be very difficult to successfully achieve with traditional vision system architectures.

The idea of high-level structure inference makes even more sense in the context of a coupled solution to recognition and segmentation. If both decisions are to be made together, representing both problems at the same level of abstraction allows the convenient expression of coupling constraints.

The work also demonstrates a novel application of Markov Random Fields in a non-homogeneous, non-isotropic, high-level application. MRFs are convenient for the representation of labelling problems, and particularly convenient for the expression of arbitrary spatial relationships that arise in the representation of spatially complex objects.

Acknowledgements

Discussions with Mike Swain and Paul Chou contributed to this work. The Institute for the Learning Sciences was established in 1989 with the support of the Arthur Andersen Worldwide Organization.

References

[Ballard, 1984] D.H. Ballard. Parameter networks. *Artificial Intelligence*, 2(1):235–267, 1984.

[Barlow, 1972] H. B. Barlow. Single units and sensation: A neuron doctrine for perceptual psychology? *Perception*, 1:371–392, 1972.

[Biederman, 1985] I. Biederman. Human image understanding: recent research and a theory. *Computer Vision, Graphics and Image Processing*, 32(1):29–73, 1985.

[Bolles, 1977] R.C. Bolles. Verification vision for programmable assembly. In *Proceedings: IJCAI-77*, pages 569–575, 1977.

[Chou, 1988] Paul B. Chou. The theory and practice of bayesian image labeling. Technical Report TR 258, Department of Computer Science, University of Rochester, August 1988.

[Cooper and Hollbach, 1987] Paul R. Cooper and Susan C. Hollbach. Parallel recognition of objects comprised of pure structure. In *Proceedings of the DARPA Image Understanding Workshop*, pages 381–391, February 1987.

[Cooper and Swain, 1989] Paul R. Cooper and Michael J. Swain. Domain dependence in parallel constraint satisfaction. In *Proceedings IJCAI-89: International Joint Conference on Artificial Intelligence*, August 1989.

[Cooper, 1988] Paul R. Cooper. Structure recognition by connectionist relaxation: Formal analysis. In *Proceedings: Conference of the Canadian Society for Computational Studies of Intelligence, CSCSI-88*, Edmonton, Alberta, June 1988.

[Cooper, 1989] Paul R. Cooper. Parallel object recognition from structure (the tinkertoy project). Technical Report 301 (Ph.D. Thesis), Dept. of Computer Science, University of Rochester, July 1989.

[Cooper, 1990] Paul R. Cooper. Parallel structure recognition with uncertainty: Coupled segmentation and matching. To appear, 1990.

[Cross and Jain, 1983] G.R. Cross and A.K. Jain. Markov random field texture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5:25–39, 1983.

[Eshera and Fu, 1986] M. A. Eshera and King-Sun Fu. An image understanding system using attributed symbolic representation and inexact graph-matching. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, PAMI-8(5), 1986.

[Feldman and Ballard, 1982] J. A. Feldman and D. H. Ballard. Connectionist models and their properties. *Cognitive Science*, 6:205–254, 1982.

[Feldman and Yakimovsky, 1974] J. A. Feldman and Yoram Yakimovsky. Decision theory and artificial intelligence: I. a semantics-based region analyzer. *Artificial Intelligence*, 5:349–371, 1974.

[Geman and Geman, 1984] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *PAMI*, 6(6):721–741, November 1984.

[Goddard *et al.*, 1988] Nigel H. Goddard, Kenton J. Lynne, and Toby Mintz. Rochester connectionist simulator. Technical Report TR233, University of Rochester, March 1988.

[Hoffman and Richards, 1986] D. Hoffman and W. Richards. Parts of recognition. In Alex P. Pentland, editor, *From Pixels to Predicates*, pages 268–294. Ablex Publishing Corporation, 1986.

[Kindermann and Snell, 1980] Ross Kindermann and J. Laurie Snell. *Markov Random Fields and their Applications.* American Mathematical Society, 1980.

[Marroquin, September 1985] Jose Luis Marroquin. Probabilistic solution of inverse problems. Technical report, MIT Artificial Intelligence Laboratory, September, 1985.

[Pentland, 1987] A. Pentland. Recognition by parts. In *Proceedings, ICCV87: First International Conference on Computer Vision*, June 1987.

[Shapiro and Haralick, 1981] Linda G. Shapiro and Robert M. Haralick. Structural descriptions and inexact matching. *IEEE-PAMI*, 3(5), 1981.

[Sher, 1987] David B. Sher. A probabilistic approach to low-level vision. Technical Report 232, Department of Computer Science, University of Rochester, October 1987.

[Swain and Cooper, 1988] Michael J. Swain and Paul R. Cooper. Parallel hardware for constraint satisfaction. In *Proceedings AAAI-88, the American Association for Artificial Intelligence Conference*, St. Paul, Minn., August 1988.

Goal-directed Smoothing for the Curvature-based Segmentation of 3-Dimensional Surfaces

Gregory Dudek and **John K. Tsotsos**
Dept of Computer Science, University of Toronto
Toronto, Canada M5S 1A4
email: dudek@ai.toronto.edu

Abstract

A new technique for describing and partitioning curved surfaces is presented. This technique is a generalization of two-dimensional "curvature-tuned smoothing" for decomposing planar curves. The three-dimensional method described here performs smoothing using the Gaussian and mean surface curvatures. As the smoothing is performed, parts are extracted in a robust manner, each at its appropriate scale. The extracted parts correspond to regions of roughly uniform curvature and constitute a rich description of the original data suitable for object recognition.

1 Introduction

In this paper, we describe a technique for computing abstracted descriptions of curved objects. A key aspect of this approach is that it reduces a dense and unmanageable image-domain description of the object to a sparse one containing the fundamental characteristics in curvature space necessary to establish the object's identity. The technique is robust in the face of noise or partially corrupt data.

The method is based of the application of an energy-based smoothing technique to the image data. In the process of producing multiple smoothed versions of the image, several different part structures are induced on the emerging representations. The segments modelled by these parts are smoothed in slightly different ways. A symbolic description of the input curve is constructed from these segments. The description allows the data to be dealt with either in terms of its coarse structure, or based on finer scale properties. Furthermore, because the representation is derived from local properties it allows recognition to proceed despite occlusion of sections of the object.

The stable extraction and measurement of curvature information in the presence of noise has been dealt with in several ways [Besl, 1988, Zucker *et al.*, 1988]. One characteristic of most existing curvature-measurement techniques is the assumption that there is a unique curvature that can be measured at each point.[1] While this is, of course, true in the analytic case, the assumption introduces significant problems for inverse problems involving noisy signals, such as those that occur in vision. Despite the respectable results that have been achieved by some researchers, the need for scale-specific operators to deal with noise problems (which also manifests itself as the need for the choice of a best smoothing scale, or the choice of an appropriate neighborhood for measurements) causes an inherent preference for certain ranges of curvature value and involves strong implicit assumptions about the underlying signal. The actual curvature of a signal depends on what we call noise and what we call signal, and hence may take on differing values depending on our goals.

One of the central ideas behind our approach is to target the smoothing technique to the particular models to be extracted. For each type of part, a smoothing operator is applied which is appropriate to this specific goal. We have previously demonstrated the advantage of this approach for two dimensional curves [Dudek and Tsotsos, 1989, Dudek and Tsotsos, 1990]; here we extend our framework to the case of smoothing for three dimensional surfaces. We conjecture that smoothing methods in other domains may also be improved upon by taking into account the particular measurements to be made from the smoothed data. In this case, curvature-tuned smoothing allows us to obtain measurements which have not been subjected to an unnatural "flattening" or distortion as a result of the smoothing.

A second key strength of our approach is that it allows patches with different curvatures to be extracted at the same location. Since curvature and scale are intimately related, this provides a multi-scale description of the object, based on curvature. Surface patches with low curvatures correspond to coarse-level aspects of object structure.

Lastly, the subdivision of curvature space obtained from our method can be made far richer than the conventional one based on merely the signs of mean and Gaussian curvature. This should allow for much more precise shape discrimination than is usually associated with curvature based object recognition. The part struc-

[1]This assumption is often only stated implicitly. Some techniques use multiple scales at an intermediate filtering stage, but then concentrate on selecting the single correct curvature at each point.

tures in terms of which the object is described includes, for example, cylindrical and spherical patches of different sizes and curvatures.

2 Method

Given the projection, $u(x,y)$, of a C^2 surface, its Gaussian curvature, K, and mean curvature, H, are given in terms of the principal curvatures k_1 and k_2 as:

$$K = k_1 k_2 \qquad H = \frac{k_1 + k_2}{2} \tag{1}$$

Note that the power of these quantities comes not only from their perceptual relevance, but also from the fact that Gaussian curvature, like the curvature of plane curves, is an intrinsic property of the curve (and hence invariant to rotation and translation). For single-valued surfaces, $z = d(x,y)$, we can define these in terms of the derivatives in the orthogonal projection plane [Do-Carmo, 1976]:

$$K(x,y) = \frac{\frac{\partial^2 u}{\partial x^2}\frac{\partial^2 u}{\partial y^2} - \frac{\partial^2 u}{\partial x \partial y}\ \frac{\partial^2 u}{\partial y \partial x}}{(1 + \frac{\partial u}{\partial x}^2 + \frac{\partial u}{\partial y}^2)^2} \tag{2}$$

$$H(x,y) = \tag{3}$$
$$\frac{1}{2}\frac{(1+\frac{\partial u}{\partial y}^2)\frac{\partial^2 u}{\partial x^2}+(1+\frac{\partial u}{\partial x}^2)\frac{\partial^2 u}{\partial y^2}-2\frac{\partial u}{\partial x}\frac{\partial u}{\partial y}\ \frac{\partial^2 u}{\partial x^2}}{(1+\frac{\partial u}{\partial x}^2+\frac{\partial u}{\partial y}^2)^{\frac{3}{2}}}$$

In practice, these quantities cannot be measured directly from real data due to its sensitivity to noise. The condition of C^2 continuity presents a further problem in that the depth map is unlikely to be continuous and that the relationship between discontinuities (edges) and noise is often a subtle one. These difficulties are usually partially addressed by the application of some non-local surface model, in particular neighborhood smoothness, to reduce the effects of high-frequency noise. The serious shortcoming of most such techniques is that the appropriate neighborhood for their application is difficult to select (and, in fact, ill-defined) and further, the smoothing itself corrupts the curvature values.[2]

One approach to dealing with real, noisy, data is to apply a minimization technique based on finding an approximating surface with pre-defined (fixed) curvatures [Dudek and Tsotsos, 1989]. In two dimensions, this "curvature-tuned smoothing" technique is based on minimizing the functional:

$$E(u) = \int_0^T \ ||u(x) - d(t)||^2 \ + \ \lambda(\kappa(u(t)) - c_i)^2 \ dt \tag{4}$$

with respect to $u(t)$, where $d(t)$ is the input curve as a function of arc length, t, and k is the curvature function. The minimization is performed for various values of the constant c_i. When this minimization is performed in conjunction with discontinuity insertion, useful decompositions of the curve can be obtained. The advantage of this method is that the series of minimizations extract structures at different curvatures, even though they may overlap arbitrarily. The parts into which the curve is decomposed correspond to uniformly curved regions which are both visually salient and useful for recognition.

[2]Typically the smoothing is posed so that it reduces the curvature. Even for ideal data, the smoothing alters the measurements.

We can generalize this two-dimensional technique to three dimensions to obtain the following (variational) minimization problem:

$$E(u) = \int_S ||u(x,y) - d(x,y)||^2 + \tag{5}$$
$$\lambda_1(k_1(x,y) - c_{1,i})^2 + \lambda_2(k_2(x,y) - c_{2,i})^2 \ dS$$

A closely related problem is minimizing:

$$E(u) = \int_S ||u(x,y) - d(x,y)||^2 + \tag{6}$$
$$\lambda_H(H(x,y) - h_i)^2 + \lambda_K(K(x,y) - k_i)^2 \ dS$$

where $d(x,y)$ is the observed data, $u(x,y)$ is the smooth surface function which constitutes a minimum of the functional, and λ_K, λ_H, h_i and k_i are constants. The information content of the Gaussian and mean curvatures is equivalent to that of the principal curvatures. An advantage of this formulation is that the signs of the Gaussian and mean curvatures are directly related to broad but useful surface classifications [Vemuri *et al.*, 1986, Besl, 1988].

The λ_K, λ_H weighting terms govern the degree of influence of the various terms in the minimization. They act as scale parameters determining both the size of the neighborhood influencing a point's value as well as the relative salience of the input data. h_i and k_i are "target values" for the curvatures which prescribe the definition of an optimal surface in terms of its principal curvatures. The minimization is performed for a variety of h_i and k_i values in what is, essentially, a sampling of curvature space. Because of the relationship between scale and curvature, the values of the λ parameters are a function of the curvature tuning.

Discontinuities must also be introduced in the process of performing this minimization. Although the *precise* placement of these discontinuities is, in general, a difficult non-linear problem [Blake and Zisserman, 1987, Terzopoulos, 1986], we need not be overly concerned with their accurate localization in all cases. The subset of the discontinuities that can be easily determined is exactly those we are most concerned with. This is because the non-linear interaction that makes discontinuities hard to place arises when discontinuities are close together and along the same orientation. This, in turn, indicates that the patch being fitted has a high local energy with respect to the current curvature parameters (since it is being broken into small pieces). Hence, discontinuities that are difficult to place correspond to regions of the input data that are poorly described by the h_i and k_i at which they were detected, and hence they are much less likely to be salient.

The image regions into which the technique naturally decomposes the image may subsequently be ranked in terms of their "energy" according to the above functional. That is, the value of the energy functional for a fixed value of the parameters h_i and k_i over a surface patch $u(x, y)$ indicate the "naturalness" or "appropriateness" of those parameters as a description of the patch. Regions with low energy correspond to natural descriptions of the underlying image. The collection of these low-energy regions constitutes a powerful description of the original image and can be used for matching. The form of the functional also assures a substantial degree of immunity to noise in the image.

3 Practical Considerations

In the vision and sensing domains, the surface function, $u(x, y)$, which gives the depth of the surface points is not a continuous function. In fact, in many practical cases such as depth from stereo, in may not be uniformly sampled. The minimization problem posed above can readily be posed in discrete terms and modified so that the data consistency term, $||u(x, y) - d(x, y)||^2$, is evaluated only at those locations where actual surface measurements are available. The use of conventional thin-plate models (equivalent to the $h_i = k_i = 0$ for almost-flat surfaces) for such surface interpolation from sparse data has been exploited with considerable success by Terzopoulos [Terzopoulos, 1986].

It is important to note the the curvature-based energy functional given by equation 7 corresponds to a large-deflection thin-plate bending model of a surface with "natural" preferred curvature. As such it is a natural model of a flexible surface being bent. On the other hand, it is also a non-linear functional.

As a result of the non-linearity nature of the system being solved, it can, in principle, admit local minima. In practice this are not of great concern for two reasons: local minima in this type of system do not appear to be physically plausible [Tauchert and Lu, 1987] and we are interested in solutions to the minization problem that occur close to the initial data, hence drastically constraining the solution space for the minimization.

4 Implementation

A variety of potential minimization techniques could be used to solve this problem. For the purposes of this implementation, a simple steepest descent method was used. The surface derivatives were computed using first-order finite differences and a discrete formulation of the energy functional in equation 7 has the following form:

$$E(u) = \sum_S \theta(x, y)(u(x, y) - d(x, y))^2 + \lambda_H(H(x, y) - h_i)^2 + \lambda_K(K(x, y) - k_i)^2 \qquad (7)$$

where $\theta(x, y)$ is zero except where there are depth measurements $d(x, y)$, where is has the value one.

It should be noted that the formulae for Gaussian and mean curvature are actually reliable only in the differential case and for C^2 surfaces. Although within each surface patch the validity of this approximation can be guaranteed after the minimization has had time to approach a minimum, the initial state of the surface, $u(x, y)$, with noisy data may be very spikey. In order for this to be smoothed out at all, it must be detected as a point of high curvature. The straightforward formulae break down, however for severe noise points (and give very low curvature estimates) [3]. As a result, it is necessary to use an augmented curvature measure that returns high curvature values at spikes or thin ridges. These augmented measures are used when the surface second derivatives are large and have the form:

$$K(x, y) = \frac{\frac{\partial^2 u}{\partial x^2}\frac{\partial^2 u}{\partial y^2} - \frac{\partial^2 u}{\partial x \partial y}\ \frac{\partial^2 u}{\partial y \partial x}}{\left(1 + \frac{\partial u}{\partial x}^2 + \frac{\partial u}{\partial y}^2\right)^2} + \phi \qquad (8)$$

$$H(x, y) = \qquad (9)$$

$$\frac{1}{2}\frac{(1+\frac{\partial u}{\partial y}^2)\frac{\partial^2 u}{\partial x^2} + (1+\frac{\partial u}{\partial x}^2)\frac{\partial^2 u}{\partial y^2} - 2\frac{\partial u}{\partial x}\frac{\partial u}{\partial y}\ \frac{\partial^2 u}{\partial x^2}}{(1+\frac{\partial u}{\partial x}^2 + \frac{\partial u}{\partial y}^2)^{\frac{3}{2}}} + \phi$$

$$\phi(u) = \begin{cases} 0 & \text{for } \nabla^2 u \leq Thrsh \\ \nabla u - sign(\nabla u) * Thrsh & \text{for } \nabla^2 u > Thrsh \end{cases} \qquad (10)$$

For surfaces of low curvature these augmented measures results in measurements close to the conventional measures (i.e. equations 2 and 4) even if they are used. For noise patches, these measures produce large "curvature" measures. One artifact of this method is that for curved surfaces at extremely oblique angles, the curvature estimate could be artificially magnified.

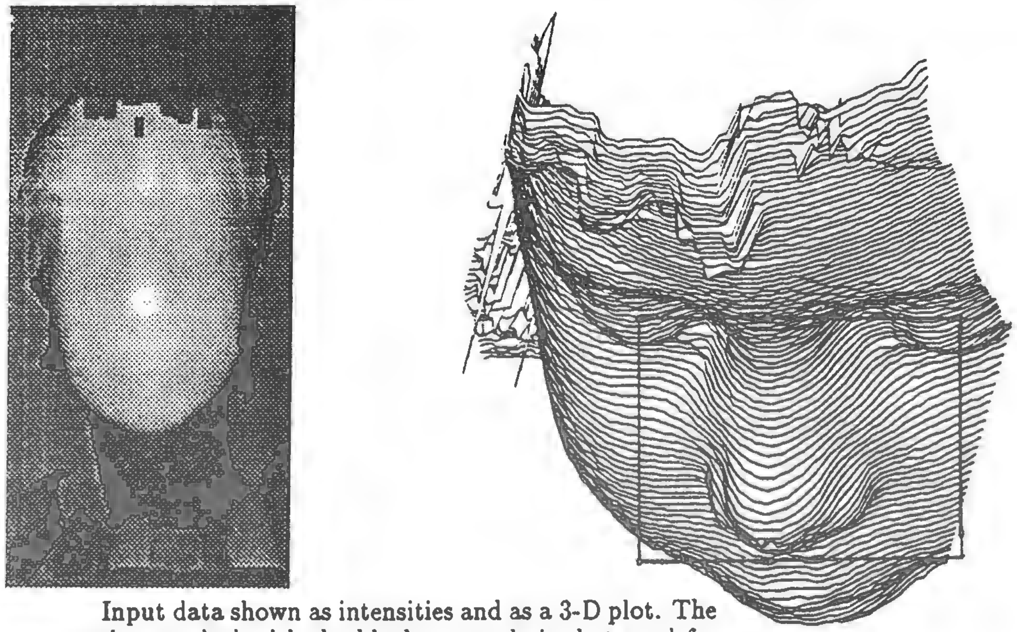
5 Experiments and Discussion

We briefly present some experimental results to illustrate the surface decomposition technique described above. A range image of a small portion of a human face (near the nose) is shown both as a 3-D perspective plot, and with depth mapped to intensity. Accompanying it are regions extracted from the image by several different curvature-tuned minimizations. The sections shown in black are those with low energy, i.e. those that were well suited to description with those combinations of curvatures.

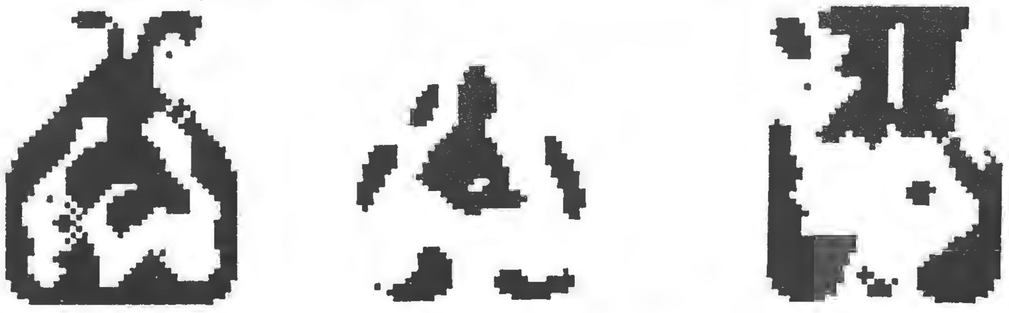
It may be interesting to reflect on the comparison between this technique and others for surface decomposition. A variety of techniques for coarse surface typing have been proposed [Marr, 1982, Vemuri *et al.*, 1986, Blicher, 1987, Koenderink and van Doorn, 1980]. In particular, some effective techniques for surface partitioning based on quadric surface classes have been developed [Ferrie and Levine, 1988, Besl and Jain, 1985, Besl, 1988]. Note, however, that these techniques have been proposed for surface decomposition into only a small set of surface classes and hence have limited discriminatory ability. This small set of surface classes, although potentially useful for coarse classification, appears to be too spare for powerful object recognition.

[3] Note that in the 2-D case, for example, at noise spikes, there may not be any potential interpolating circle that allows a single-valued circular function to be interpolated through the data. In short, the conventional conception of curvature in this context becomes ill-defined.

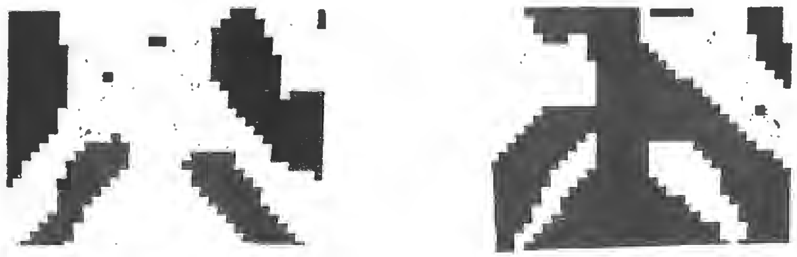
Figure 1: Input image and sample decompositions for different curvature tunings. Observe that the sides of the nose and the cheeks are selected by the smoother tuned for flat surfaces. The cylindrical tuning picks out the bridge of the nose, the edges of the nostrils, and the edges of the eye sockets. The spherical tuning selects the tip of the nose. In practice, these data would be postprocessed you select only certain "canonical" descriptors at each level. The curvature tuning was quite coarse; a finer sampling of curvature space would produce fewer, more precise, descriptors at each level.



Input data shown as intensities and as a 3-D plot. The region marked with the black rectangle is that used for the computations below.



Convex cylindrical tuning Convex spherical tuning Planar tuning



Detail of nose showing concave cylindrical regions (in black) at sides of nose and nostrils (left) and convex cylindrical regions down center of nose and around edges (right).

Perhaps more important, these methods implicitly use a scale-specific decomposition operation and allow only one descriptive part to cover any given part of the image. The technique described here can be used to produce either coarse descriptors as well as richer descriptive elements. Finally, the technique proposed here includes segmentation as a natural part of the measurement process rather than imposing it as a arbitrary second stage of processing.

6 Summary

A technique for "smoothing" data using a family of curvature-tuned minimization procedures has been introduced. The application of this technique divides the image into regions in a robust manner. These regions are defined by roughly uniform properties in terms of the surface curvatures. This decomposition captures structure at different curvatures (and scales) even if they occur at the same location.

The resulting description appears to be appropriate for a variety of recognition tasks. A similar technique has been found to be useful for recognition in the domain of plane curves. Further issues alluded to, but not discussed here, are the detection of discontinuities, the completeness of the representation, and the robustness of the description; these issues are currently being explored.

7 Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council and through the Information Technology Research Center, a Province of Ontario Center of Excellence. The second author is a Fellow of the Canadian Institute for Advanced Research.

The authors are grateful to Allan Jepson, John Scherk and Demitri Terzopoulos their helpful comments.

References

[Besl and Jain, 1985] Paul J. Besl and Ramesh C. Jain. Three-dimensional object recognition. *computing surveys*, 17(1):75–145, March 1985.

[Besl, 1988] Paul J. Besl. *Surfaces in Range Image Understanding.* Springer-Verlag, New York, N.Y., 1988.

[Blake and Zisserman, 1987] Andrew Blake and Andrew Zisserman. *Visual Reconstruction.* MIT Press, Cambridge, Mass., 1987.

[Blicher, 1987] Peter A. Blicher. A shape representation based on geometric topology: Bumps, gaussian curvature, and the topological zodiac. *International Joint Conference of Artificial Intelligence*, pages 767–770, August 1987.

[DoCarmo, 1976] Manfredo P. DoCarmo. *Differential Geometry of Curves and Surfaces.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1976.

[Dudek and Tsotsos, 1989] Gregory Dudek and John K. Tsotsos. Using curvature information in the decomposition and representation of planar curves. *NATO Advanced Study Institute on Robotics and Active Vision*, July 1989.

[Dudek and Tsotsos, 1990] Gregory Dudek and John K. Tsotsos. Recognizing planar curves using curvature-tuned smoothing. *10th International Conference on Pattern Recognition*, pages MS–1041, June 1990.

[Ferrie and Levine, 1988] F. P. Ferrie and M. D. Levine. Deriving coarse 3d models of objects. *Computer Vision, Graphics and Image Processing Computer Vision and Pattern Recognition*, pages 345–353, June 1988.

[Koenderink and van Doorn, 1980] J. J. Koenderink and A. J. van Doorn. Photometric invariants related to solid shape. *Optica Acta*, 27(7):981–996, 1980.

[Marr, 1982] David Marr. *Vision.* W. H. Freeman and Co., New York, 1982.

[Tauchert and Lu, 1987] T. R. Tauchert and W. Y. Lu. Large deformation and postbuckling behavior of an initially deformed rod. *Journal of Non-Linear Mechanics*, 22(6):511–520, 1987.

[Terzopoulos, 1986] Demitri Terzopoulos. Regularization of inverse visual problems involving discontinuities. *Pattern Analysis and Machine Intelligence*, 8(4):413–424, 1986.

[Vemuri *et al.*, 1986] B. C. Vemuri, A. Mitiche, and J. K. Aggarwal. Curvature-based representation of objects from range data. *Image and vision computing*, 4(2):107–114, May 1986.

[Zucker *et al.*, 1988] Steven W. Zucker, Chantal David, Allan Dobbins, and Lee Iverson. The organization of curve detection: Coarse tangent fields and fine spline coverings. *Proceedings of the 2nd ICCV*, pages 568–577, Dec. 1988.

Can We Build Learning Robots?
Peut-on construire des robots qui apprennent?

Tom M. Mitchell
Department of Computer Science and Robotics
Carnegie Mellon University
Pittsburgh, Pennsylvania, U.S.A.

Abstract

Yes. At least simple ones...

One approach to overcoming the brittleness and inflexibility of current robot systems is to develop approaches that allow them to learn from their experience. This talk discusses the general problem of robot learning, and focuses on three types of learning that are currently exhibited by prototype robot systems in our laboratory. The first is learning of strategies for approaching and recognizing simple objects (e.g., cups, boxes) using sonar sensors aboard a mobile robot. The robot is trained by showing it a new type of object, and it inductively learns to approach, sense, and discriminate the object from others it has observed. The second type of learning automatically improves robot reactivity for routine actions by compiling the results of slow, deliberate planning into stimulus-response rules. These learned rules produce the same decisions as the original planning process, but reduce the reaction time of the robot from minutes to subsecond. The third type of learning improves the robot's ability to correctly predict the effects of its actions on the world, thus improving the correctness of its control decisions. This third project uses a robot arm manipulating objects, with feedback provided by a vision system.

The talk will summarize current results from these three learning robot systems, and consider some of the MANY open issues that remain.

Oui. Du moins s'ils sont simples...

Une approche pour venir à bout de la fragilité des systèmes robotiques actuels est de développer des mécanismes qui leur permettent d'apprendre par expérience. Cette présentation discute le problème général d'apprentissage en robotique et présent trois types d'apprentissages qui sont actuellement utilisés par des prototypes de robots dans notre laboratoire. Le premier est l'apprentissage de stratégies pour approcher et reconnaitre de simples objets (par exemples des boites, des verres) à l'aide de senseurs soniques montés sur un robot mobile. L'apprentissage se fait en présentant un nouveau type d'objet au robot qui apprend par induction à l'approcher, le reconnaitre et le différencier d'autres objets qu'il a observés. Le second type d'apprentissage améliore la vitesse de réaction du robot pour des actions routinières en compilant les résultats d'une planification intentionnelle lente en une série de règles stimulus-réaction. Ces règles produisent les mêmes décisions que le processus original, mais réduisent le temps de réaction du robot de plusieurs minutes à moins d'une seconde. Le troisième type d'apprentissage améliore la capacité qu'a le robot de prédire les effet de ses actions sur le monde, améliorant ainsi la qualité de ses décisions de contrôle. Ce troisième project utilise un bras mécanique et un système de vision pour manipuler des objets.

Cette présentation résume des résultats récents obtenus avec ces trois systèmes de robots et considère quelques unes des NOMBREUSES questions qui demeurent.

Planning the Future of Natural Language Research (even in Canada)

Graeme Hirst
Department of Computer Science
University of Toronto
Toronto, Ontario, CANADA

Abstract

I believe that the following issues will (or should) become important in AI research in natural language in the 1990s:

- Nuances of language and style;
- Problems in knowledge representation for natural language;
- Deciding when partial understanding suffices.

I will discuss each of these, describing the present foundations and why I think the topics will be important for the future. In addition, I will discuss the role of NL research in Canada, especially with regard to machine translation.

Obtaining Colour Signal Spectra for Colour Constancy

Brian V. Funt, Mark S. Drew, and Jian Ho
School of Computing Science
Simon Fraser University
Vancouver, B.C.
Canada V5A 1S6
604-291-3126
e-mail funt@cs.sfu.ca

Abstract

We wish to report on progress on two problems related to colour constancy. The first is how to obtain information about the colour signal from a standard colour image; and the second is how to extract the illumination and reflectance components from a colour signal. In particular, we will show how colour signal information can be extracted from either chromatic aberration or mutual illumination and then used for colour constancy.

Chromatic aberration blurs an image in a wavelength-dependent manner. By analysing the chromatic-aberration-induced blur at a reflectance edge, the difference in the colour signals from the regions forming the edge can be calculated. Mutual illumination between surfaces (i.e. light reflected off surface A impinging on surface B) also contains clues about the respective colour signals from A and B. The light a camera receives from a point on A where mutual illumination from B is absent tells us the colour of one component of the illumination incident on B; namely, that part reflected from A onto B. Comparing a point on B where mutual illumination is present to one where it is absent allows the ambient illumination to be determined. The method is formalized in terms of finite-dimensional models of light and surface reflectance and leads to a set of simultaneous, non-linear equations.

The algorithm for decomposing a colour signal minimizes the difference between it and approximations to it formed as products of finite-dimensional models of potential illuminants and surface reflectances. The method succeeds because the statistical properties of typical illuminants such as daylight differ from those of typical surface reflectances.

1 Introduction

The spectrum of light reflected from a surface contains information about both the surface's reflectance and the spectrum of the incident illumination. In general, it would be very useful to be able to decompose the spectrum of the reflected light, known as the 'colour signal,' into two components– one due to the surface reflectance and a second due to the illumination. Such a decomposition would yield the colour of the surface independent of the colour of the illumination. Since surface colour is properly a surface property, not a property of reflected light, it must be encoded with a descriptor that remains constant despite changes in the colour of the illumination. The surface reflectance component of the colour signal is one such descriptor.

Figure 1 depicts the situation. A point on a surface with percent spectral reflectance $S(\lambda)$ illuminated with light of spectral power distribution (SPD) $E(\lambda)$ reflects light, the *colour signal*, of SPD $C(\lambda) = E(\lambda) \times S(\lambda)$. Ideally, we would like to obtain $S(\lambda)$ from some type of sampled measurement of the colour signal, $C(\lambda)$. The traditional colour camera samples the colour signal in three broad bands defined by three sensor response functions, $R_k(\lambda)$, $k = 1..3$. The camera output is defined by:

$$\rho_k = \int C(\lambda)\, R_k(\lambda)\, d\lambda\,, \quad k = 1..3 \qquad (1)$$

While less general than the complete surface reflectance function, a second common type of colour-constant descriptor is defined as what the camera output would be if the surface were illuminated under a standard, white illuminant such as D65. In the case of an 'ideal white' where $E(\lambda) = 1$, the colour-constant descriptor is simply:

$$\rho_k = \int S(\lambda)\, R_k(\lambda)\, d\lambda\,, \quad k = 1..3 \qquad (2)$$

Many approaches to colour constancy [Maloney and Wandell, 1986; Land, 1974; Gershon *et al.*, 1987] take the view that a set of output camera vectors $\vec{\rho}$ obtained from one or more points constitutes all that is known about the colour signal reflected from them. More than this, however, can be established by taking into account the effects of chromatic aberration and mutual illumination, as long as the camera parameters are known and some assumptions about the scene are valid. With more information about the colour signal in hand, a different approach can be taken to calculating colour-constant descriptors.

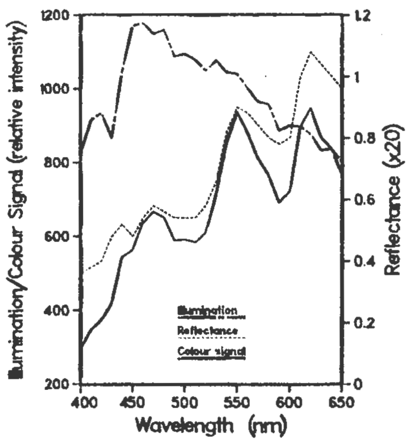


Figure 1: Incident illumination and reflectance combine to form the colour signal. Shown here is the colour signal resulting from standard daylight D65 illuminating Krinov reflectance # 53 (heather, dense growth before flowering)[Krinov, 1947].

2 Mutual Illumination

Mutual illumination causes much of the illumination variation in a scene and therefore contributes to the need for a colour constancy algorithm. It occurs when light reflected from one surface hits a second surface thus illuminating it with a different SPD than that of whatever direct illumination is present. As a concrete example, consider a red and a blue surface meeting at a concave edge outdoors in the sun. The basically white mixture of sunlight and skylight impinges on the red surface from which it reflects as dominantly red. Illuminating the blue surface, therefore, is a mixture of reflected red and direct white light, the relative mixture of which varies across the surface. By symmetry, reflected blue light also impinges on the red surface, which while very damped, again will bounce back to the blue side.

For two mutually illuminating surfaces A and B, let $C^A_{\mathrm{m}}(\lambda)$ be the colour signal reflected from surface A at a location where mutual illumination from B is present and $C^A(\lambda)$ be the colour signal where mutual illumination is not present; and similarly $C^B_{\mathrm{m}}(\lambda)$ and $C^B(\lambda)$. The fraction of $C^A(\lambda)$ striking B is denoted by α_{AB}. If we ignore multiple reflective bounces (see [Drew *et al.*, 1989]) then to first order we have:

$$C^A_{\mathrm{m}}(\lambda) = C^A(\lambda) + \alpha_{BA}\, C^B(\lambda)\, S^A(\lambda) \qquad (3a)$$

$$C^B_{\mathrm{m}}(\lambda) = C^B(\lambda) + \alpha_{AB}\, C^A(\lambda)\, S^B(\lambda) \qquad (3b)$$

If we could measure colour signal SPDs directly, then these equations could be used to solve for the percent reflectance up to an overall multiplicative constant. For example, using (3a)

$$\alpha_{BA}\, S^A(\lambda) = [C^A_{\mathrm{m}}(\lambda) - C^A(\lambda)]/C^B(\lambda) \qquad (4)$$

If colour signal SPDs are not available, then we can employ finite dimensional linear models to approximate the illumination and surface reflectances in a manner similar to [Maloney and Wandell, 1986; Sallstrom, 1973; Buchsbaum and Goldstein, 1979]. Using appropriate basis functions, E_i, $i = 1..m$ for illumination and S_j, $j = 1..n$ for reflectance, models of low dimensionality approximate typical illuminations, such as daylight, and reflectances of natural objects quite well[Maloney, 1986]. This technique can be extended to model colour signals[Drew and Funt, 1990] using basis functions $C_l(\lambda)$ derived by principal components analysis of a large set of measured colour signals.

In terms of a finite-dimensional linear model with n basis functions, a colour signal is approximated:

$$C(\lambda) = \sum_{l=1}^{n} \gamma_l\, C_l(\lambda) \qquad (5)$$

with weights γ_l.

The basis functions are fixed, so it is the weights γ_l that represent the colour signal. Substituting the finite-dimensional expansion into the camera response equation (1) yields

$$\rho_k = \sum_{l=1}^{n} \gamma_l \int C_l(\lambda)\, R_k(\lambda)\, d\lambda \qquad (6)$$

Given a set of camera response measurements ρ_k, this equation set can be solved for the weights γ_l that represent an approximation to the colour signal $C(\lambda)$. For 'natural' objects (e.g. leaves or rose petals) lit by natural illuminants (e.g. sunlight), the recovery of the colour signal is surprisingly good even for $n = 3$.

In the case of mutual illumination, three colour signals are required to determine the surface reflectance of A: two from points on surface A, one within the mutual illumination area and one outside it, and a point from B unaffected by mutual illumination from A. Approximations to these colour signals can be obtained from the camera responses at these locations via equation (6) and employed in (4) to establish the surface reflectance and colour of A.

In [Drew *et al.*, 1989] an alternative set of equations is presented in which the surface reflectance is solved for directly without first determining the three colour signals.

3 Chromatic Aberration

Chromatic aberration is a wavelength-dependent distortion and as such it contains information about an incoming colour signal. A single-element lens, such as that of the eye, only can focus one wavelength at a time. All other wavelengths will be out of focus. Another way of saying this is that the focal length of a lens is a function of the refractive index of its material, and in general, the refractive index of a typical material varies monotonically with wavelength. A glass prism illustrates this.

Given the wavelength-dependent nature of chromatic aberration, what information can be gleaned from it about the colour signal? To answer this question, we first consider a model of image formation in the presence of chromatic aberration. The point spread function (PSF) characterizes a shift-invariant linear system, such as that of an ideal lens, by specifying the system's output in response to an idealized point impulse input. The image a lens forms is then described by the convolution of the PSF with the input, the input being the pattern of light entering the lens.

In the case of chromatic aberration, the PSF for a single wavelength λ is simply that of a defocussed lens and the image for that wavelength alone is the convolution of that PSF with the input. For a second wavelength λ', the PSF is again that of a defocussed lens, but with a different degree of defocussing. The defocussed images corresponding to varying wavelengths superimpose additively to create one pattern of light intensity on the image plane.

The geometrical (i.e. ignoring diffraction) PSF of a defocussed lens is a circularly symmetric 'pillbox' function of radius $R(\lambda)$:

$$h(r,\lambda) = \begin{cases} 0, & r > R(\lambda) \\ 1/\pi R^2, & r \leq R(\lambda) \end{cases} \tag{7}$$

The chromatic-aberration-induced defocussing depends on wavelength, as is made explicit by expressing R as a function of λ. Figure 2 shows how the image of a single point is the superposition of a set of disks defined by the wavelength-dependent PSFs. A crucial observation is that the thickness of each disk varies both as a function of the height PSF **and** of the energy in the colour signal $C(\lambda)$ emanating from the point for the relevant wavelength λ.

Now consider the image of a scene. From each image location (x,y) there is a corresponding colour signal, so the scene can be described as a function $C(x,y,\lambda)$. The grey level, intensity image (i.e. black and white picture) in the presence of chromatic aberration is given by

$$I(x,y) = \int_{\text{visible}} C(x,y,\lambda) \star h(r,\lambda)\, d\lambda \tag{8}$$

where '$\star$' is the convolution operator.

What will the resulting image look like for a reflectance edge occurring between two regions of different colour? Suppose, for the sake of having names to use, the regions are red and blue and the transition from red to blue is a vertical step function. In the red region far from the edge, the convolution simply blurs red into red creating no change. Near the edge, however, some of the red will spread into the blue and vice versa. Such a point's grey level intensity can be expressed as a baseline intensity due to the red, less the amount of red blurred into the blue region, plus the amount of blue blurred into the red region.

For the reflectance edge case, this relative blurring can be made precise. The constraints on C allow the integral equation (8) to be solved for the difference (the DSPD)

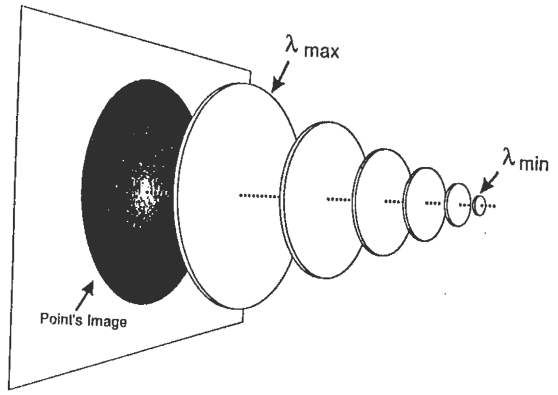


Figure 2: The image of a single point is the superposition of a set of disks defined by the wavelength-dependent PSFs.

between the SPD of the colour signal from the red region and that of the blue. Details are given in [Funt and Ho, 1988], but intuitively it is clear by the symmetry of the blurring between the red and blue regions that it will only be possible to obtain the colour signal of one relative to the other.

The influence of diffraction on the PSF cannot be ignored for the DSPD extraction to work properly. Figure 3 shows two examples of PSFs for different defocussings, which clearly bear only superficial resemblance to the often-suggested approximation (e.g., [Horn, 1986, p. 127]) the pillbox. These PSFs were calculated from the inverse Fourier transform of the optical transfer functions given in [Stokseth, 1969].

Figure 4 plots intensity profiles across a unit step edge convolved with different PSFs. Profiles generated using the pillbox PSFs are shown for comparison. The intensity profile across a reflectance edge between two arbitrarily coloured regions will be a weighted sum of a set of such profiles, with weights proportional to the DSPD at the corresponding wavelength. We are currently testing a new chromatic aberration algorithm based on this observation that fits the reflectance-edge profile as a sum of the PSF-convolved edge profiles, thereby solving for the weights representing the DSPD.

4 Separating Illumination from Reflectance

An important premise of the above is that knowing the SPD of a colour signal or the DSPD of two colour signals will be of some use. In particular, can the SPD $C(\lambda)$, which is the product of two functions $E(\lambda)$ and $S(\lambda)$, be

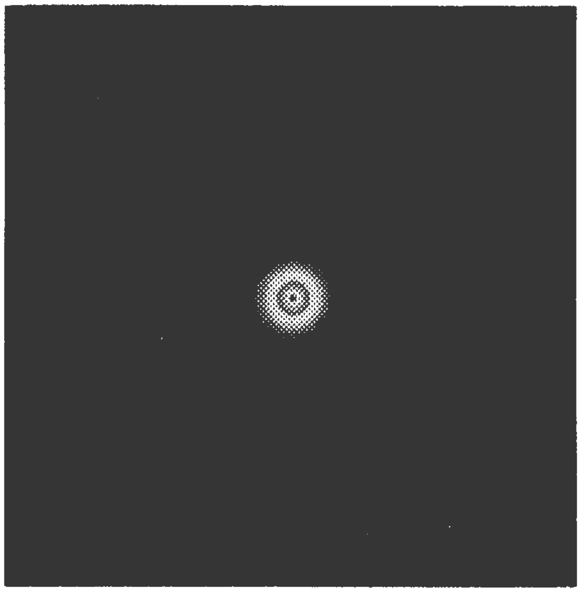
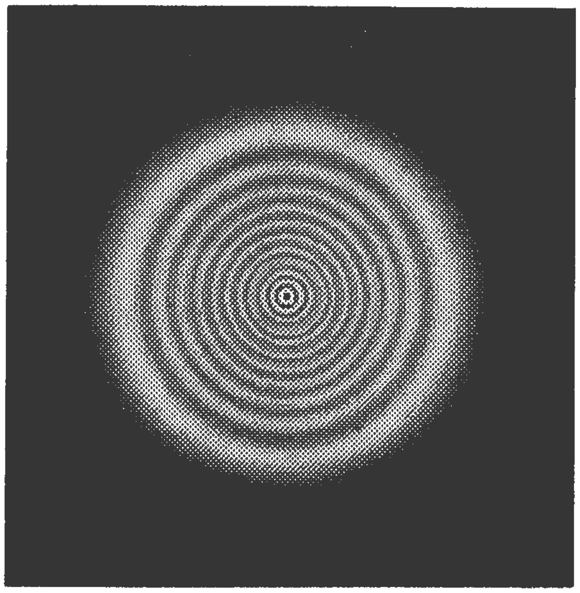



Figure 3: For a system focussed at 350nm, these are the PSFs for 360nm and 410nm.

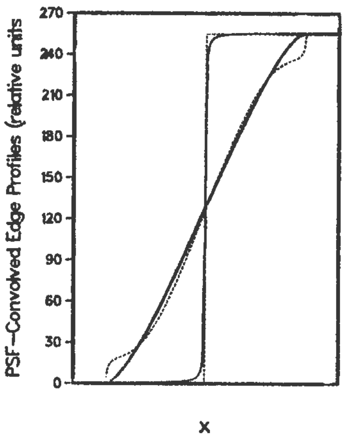


Figure 4: Step-edge profiles using zero defocussing and large defocussing including the effects of diffraction (solid lines) and the corresponding no-diffraction profiles (dashed lines).

separated into its respective components? It might at first appear that the answer is negative because of the product nature of their intermixture in the colour signal. Typical illuminants and typical reflectances, however, differ in their statistical properties.

Finite-dimensional linear models of illumination and reflectance could be based on any linearly independent set of basis functions, but for models of low dimensionality to succeed, the best bases can be found via a principal components analysis[Jolliffe, 1986] of large and representative sets of illumination and reflectance measurements. The basis functions so obtained capture the statistically most relevant features of the data. The bases determined for illumination and reflectance by Judd [Judd *et al.*, 1964] and Cohen [Cohen, 1964] as well as the bases determined from our own analysis of the reflectance data of Krinov [Krinov, 1947] have the interesting property that the set of E_iS_j pairs, ($i = 1,..4$; $j = 1..6$) is linearly independent. As a consequence, the finite-dimensional approximations to the reflectance and illumination components can be extracted from the colour signal.

The colour signal components are found by minimizing the least-squares difference between the actual colour signal and synthesized colour signals composed as products of candidate illuminations and reflectances— the candidates being those illuminations and reflectances that can be generated using the basis functions established by the principal component analyses. The quantity to minimize is

$$\sum_{l=0}^{v}\left\{\left[\sum_{i=1}^{m}\epsilon_i E_i(\lambda_l)\right]\left[\sum_{j=1}^{n}\sigma_j S_j(\lambda_l)\right] - C(\lambda_l)\right\}^2 . \tag{9}$$

Setting the partial derivatives with respect to the unknowns to zero leads to the corresponding 'normal' equations to be solved for ϵ_i and σ_j.
For $k = 1$ to m,

$$\sum_{l=0}^{v} E_k(\lambda_l)\left[\sum_{j=1}^{n}\sigma_j S_j(\lambda_l)\right]\left\{\left[\sum_{i=1}^{m}\epsilon_i E_i(\lambda_l)\right]\left[\sum_{j=1}^{n}\sigma_j S_j(\lambda_l)\right] - C(\lambda_l)\right\} = 0 .$$

For $k = 1$ to n,

$$\sum_{l=0}^{v} S_k(\lambda_l)\left[\sum_{i=1}^{m}\epsilon_i E_i(\lambda_l)\right]\left\{\left[\sum_{i=1}^{m}\epsilon_i E_i(\lambda_l)\right]\left[\sum_{j=1}^{n}\sigma_j S_j(\lambda_l)\right] - C(\lambda_l)\right\} = 0 . \tag{10}$$

Tests on the accuracy of the recovered illumination spectra and surface reflectance functions show quite good results and a theorem ties the accuracy of the recovery to the accuracy of the underlying finite-dimensional model[Ho *et al.*, 1990]. The recovery is unique up to an overall multiplicative scale factor representing the relative brightness of the illumination. For a colour signal which is the product of an illumination and reflectance that can each be *exactly* expressed in terms of the basis functions, the recovery is exact.

When a DSPD is available, not an SPD, then the above equations can be solved for the ϵ_i and terms $\Delta\sigma_j$ representing the difference in the reflectance between the two regions. Once the illumination weights ϵ_i are known, the reflectance weights σ_j for each side of an edge can be determined from illumination-corrected responses of a colour camera. Thus colour-constant descriptors are recovered for each region.

5 Conclusion

We have followed on the work of [Sallstrom, 1973], who first proposed approximating spectra by finite-dimensional linear models, as well as that of [Maloney and Wandell, 1986; Wandell, 1987] where they are exploited so cleverly in a computational model of colour constancy. The advantage of finite-dimensional models lies in the fact that they allow one to think about, represent and manipulate whole spectra very easily— a significant advance over representing all colour information in terms of a vector of sensor responses.

Once one makes spectra one's focus instead of trichromatic sensor responses, it becomes natural to seek ways of obtaining more information about them. In terms of the spectrum of the incoming colour signal, one option is to measure it directly with a spectrometer or, as we have shown, chromatic aberration and mutual illumination effects can be analysed to provide knowledge of some of its characteristics. In either case, additional information about the colour signal helps establish its surface reflectance and illumination components, which are at the heart of the problem of producing colour-constant surface descriptors.

6 Acknowledgements

We wish to acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada and the Centre for Systems Science, Simon Fraser University.

References

[Buchsbaum and Goldstein, 1979] G. Buchsbaum and J.L. Goldstein. Optimum probabalistic processing in colour perception. ii. Colour vision as template matching. *Proc. R. Soc. Lond. B.*, 205:249–266, 1979.

[Cohen, 1964] J. Cohen. Dependency of the spectral reflectance curves of the Munsell color chips. *Psychon. Sci.*, 1:369–370, 1964.

[Drew and Funt, 1990] M.S. Drew and B.V. Funt. Recovering the best colour signal from an RGB signal. Technical Report CSS/LCCR TR 90-01, Simon Fraser University School of Computing Science, 1990.

[Drew *et al.*, 1989] M.S. Drew, J. Ho, and B.V. Funt. Color constancy from mutual reflectance. Technical Report CSS/LCCR TR 89-02, Simon Fraser University School of Computing Science, 1989.

[Funt and Ho, 1988] B.V. Funt and J. Ho. Color from black and white. In *Proceedings of the Second International Conference on Computer Vision, Tarpon Springs Dec 5-8*, pages 2–8. IEEE, 1988. and *Int. J. Computer Vision*, Vol. 3 1989, pp. 109-117.

[Gershon *et al.*, 1987] R. Gershon, A.D. Jepson, and J. Tsotsos. From [R,G,B] to surface reflectance: Computing color constant descriptors in images. In *Proceedings of IJCAI conference*, 1987.

[Ho *et al.*, 1990] J. Ho, B.V. Funt, and M.S. Drew. Separating a color signal into illumination and surface reflectance components: Theory and applications. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 1990. In press.

[Horn, 1986] B.K.P. Horn. *Robot Vision*. MIT Press, 1986.

[Jolliffe, 1986] I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[Judd *et al.*, 1964] D.B. Judd, D.L. MacAdam, and G. Wyszecki. Spectral distribution of typical daylight as a function of correlated color temperature. *J. Opt. Soc. Am.*, 54:1031–1040, 1964.

[Krinov, 1947] E.L. Krinov. Spectral reflectance properties of natural formations. *Technical Translation TT-439, National Research Council of Canada*, 1947.

[Land, 1974] E.H. Land. The retinex theory of color vision. *Proc. R. Inst.*, 47:23–58, 1974.

[Maloney and Wandell, 1986] L.T. Maloney and B.A. Wandell. Color constancy: a method for recovering surface spectral reflectance. *J. Opt. Soc. Am. A*, 3:29–33, 1986.

[Maloney, 1986] L.T. Maloney. Evaluation of linear models of surface spectral reflectance with small numbers of parameters. *J. Opt. Soc. Am. A*, 3:1673–1683, 1986.

[Sallstrom, 1973] P. Sallstrom. Colour and physics; some remarks concerning the physical aspects of human colour vision. Technical report, University of Stockholm: Institute of Physics Report 73-09, 1973.

[Stokseth, 1969] P.A. Stokseth. Properties of a defocused optical system. *J. Opt. Soc. Am.*, 59:1314–1321, 1969.

[Wandell, 1987] B.A. Wandell. The synthesis and analysis of color images. *IEEE Trans. Patt. Anal. and Mach. Intell.*, PAMI-9:2–13, 1987.

Author's Index
Répertoire des auteurs