ACTES
SIXIÈME CONFÉRENCE CANADIENNE SUR L'INTELLIGENCE ARTIFICIELLE

PROCEEDINGS
SIXTH CANADIAN CONFERENCE ON ARTIFICIAL INTELLIGENCE

Commanditée par: La Société canadienne pour l'étude de l'intelligence par ordinateur

Sponsored by: Canadian Society for Computational Studies of Intelligence

ÉCOLE POLYTECHNIQUE DE MONTRÉAL
MONTRÉAL QUÉBEC CANADA
21-23 mai/May 1986

ROBERT HOLTE
TOWER C
BRUNEL UNIVERSITY
UXBRIDGE
ENGLAND UB8 3PH

ACTES

SIXIEME CONFERENCE CANADIENNE SUR
L'INTELLIGENCE ARTIFICIELLE

PROCEEDINGS

SIXTH CANADIAN CONFERENCE ON
ARTIFICIAL INTELLIGENCE

Commanditée par:
La Société canadienne pour l'étude de l'intelligence par ordinateur

Sponsored by :
Canadian Society for Computational Studies of Intelligence

ÉCOLE POLYTECHNIQUE DE MONTRÉAL
MONTRÉAL QUÉBEC CANADA

21 - 23 mai / May 1986

ISBN 2-7605-0409-3

Dépôt légal — 2e trimestre 1986
Bibliothèque nationale du Québec
Bibliothèque nationale du Canada
Imprimé au Canada

MESSAGE DU PRÉSIDENT DE LA CONFÉRENCE

Il me fait plaisir de souhaiter la bienvenue à tous les participants à la sixième conférence canadienne sur l'intelligence artificielle commanditée par la Société canadienne pour l'étude de l'intelligence par ordinateur (SCEIO).

L'organisation de cette conférence est un bel exemple de coopération entre les universités montréalaises, incarné dans le Centre de recherche informatique de Montréal inc.(CRIM).

L'équipe de Montréal a travaillé en collaboration avec le comité de programme présidé par le Dr Bill Havens, les officiers de la SCEIO et son président le Dr Gordon McCalla, ainsi qu'avec le rédacteur en chef du bulletin canadien sur l'intelligence artificielle, le Dr Graeme Hirst.

Je désire remercier les organismes suivants pour leur aide :

- Le Centre de recherche informatique de Montréal
- Le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG)
- L'Institut canadien des recherches avancées
- L'École Polytechnique de Montréal

J'aimerais également offrir mes plus sincères remerciements aux personnes suivantes :

- Aux membres du comité organisateur pour avoir gracieusement fait don de leur temps à l'organisation de cette conférence.

- Au Dr J. Charles Giguère, directeur général du CRIM, pour avoir mis à notre disposition les ressources humaines et matérielles du CRIM.

- Au Dr Roland Doré, directeur de l'École Polytechnique de Montréal, pour l'accueil et le soutien de cette conférence.

- Au Pr Claude Dubeau, directeur du service de l'Éducation permanente de l'École Polytechnique de Montréal et les membres de son service, pour l'organisation matérielle.

- À Mlle Nadine Lasalle, pour son enthousiasme et son dévouement dans l'organisation de plusieurs aspects de la conférence.

Renato De Mori

Président de la conférence

MESSAGE FROM THE GENERAL CHAIRMAN

It is a great pleasure to welcome all participants to the 1986 Conference of the Canadian Society for the Computational Studies of Intelligence.

The organization of this Conference represents a beautiful example of cooperation among all the Montréal area universities, facilitated by the centre they formed as a consortium, the Centre de recherche informatique de Montréal inc. (CRIM).

The team in Montreal interacted well with the programme committee chaired by Dr. Bill Havens, with officers of the CSCSI and its President Dr. Gordon McCalla, and with the Editor of Canadian Artificial Intelligence Newletters, Dr. Graeme Hirst.

I am pleased to acknowledge support from :

- Centre de recherche informatique de Montréal inc. (CRIM)
- The Natural Sciences and Engineering Council of Canada (NSECC)
- The Canadian Institute for Advanced Reaserch (CIAR)
- École Polytechnique de Montréal

I would like to express my warmest thanks to :

- The members of the Organizing Committee for the time they generously spent on this conference;

- Dr. J. Charles Giguère, Director of CRIM, for making CRIM's excellent physical and human resources available to the Organizing Committee;

- Dr. Roland Doré, Director of l'École Polytechnique de Montréal, for his welcome and support for this conference;

- Prof. Claude Dubeau, Director of l'Éducation permanente, at l'École Polytechnique de Montréal and his team for the material organization.

- Ms. Nadine Lasalle, for the enthousiasm and dedication she has brought to her tasks as Conference Secretary.

Renato De Mori

General Chairman

MOT DE BIENVENUE DU PRESIDENT DU PROGRAMME

A titre de président du comité du programme de la Conférence nationale canadienne d'intelligence artificielle, j'aimerais vous souhaiter la bienvenue à cette conférence et la bienvenue dans cette très belle ville qu'est Montréal. Cette conférence est la sixième conférence bisannuelle organisée par la SCEIO et nous sommes fiers du programme technique présenté dans les actes. La SCEIO a acquis la réputation d'organiser des rencontres internationales informelles où des recherches importantes en intelligence artificielle peuvent être présentées et discutées. La conférence 1986 s'inscrit dans cette tradition. Nous avons reçu plus de 100 propositions de communications provenant d'Amérique du nord, d'Europe et d'Asie. Les membres du comité du programme et ceux du jury ont été ravis, mais aussi débordés. Nous avons pu composer un programme qui regroupe des communications de la plus haute qualité. Néanmoins, plusieurs bonnes communications ont dû être rejetées afin de rendre la taille du programme plus raisonnable et de conserver le style informel de la conférence. En effet, nous avons réservé des périodes en dehors des moments de présentation des communications pour permettre aux participants de se rencontrer et de discuter de leurs travaux.

Je voudrais profiter de l'occasion pour remercier certaines personnes qui ont participé à la réalisation du programme technique. Sans leur travail bénévole, la conférence n'aurait pas pu avoir lieu. Je veux d'abord remercier le comité du programme qui a été responsable du travail de sélection. Il a traité de bon coeur le grand nombre de propositions de communications sans penser à la charge de travail supplémentaire que cela représentait. De plus, le déluge des propositions de communication m'a obligé à faire appel au dernier moment à de nouveaux membres pour augmenter le comité de programme. Je les remercie d'avoir accepté cette tâche à si brève échéance. Les arbitres qui ont lu les propositions de communication méritent aussi notre reconnaissance. L'évaluation critique des travaux d'autres chercheurs est une tâche extrêmement difficile. Je vous remercie tous pour les commentaires que vous avez communiqués rapidement. Finalement, nous remercions toutes les personnes qui ont oeuvré à des tâches moins visibles mais toutes aussi essentielles telles que la préparation de l'appel aux communications, la conception de la publicité, la réception et la distribution des manuscrits, la correspondance avec les auteurs et le comité du programme et l'exécution de tâches peu reconnues mais essentielles. Ces personnes sont Kathy Findler, Nadine Lasalle, Sandy Sapers, May Vink et Linsey Wey.

Vous trouverez dans les actes le fruit de notre travail. Nous osons espérer que les résultats de recherche communiqués ici contribueront de façon significative à l'avancement du domaine de l'intelligence artificielle. Profitez-en!

Bill Havens

Président du comité du programme

WELCOME FROM THE PROGRAM CHAIRMAN

As program chairman for the 1986 Canadian National Artificial Intelligence Conference, let me welcome you to the conference and to the wonderful city of Montreal. This is the sixth biennial conference sponsored by the CSCSI/SCEIO and we are very proud of the technical program appearing in these proceedings. CSCSI has developed a reputation for conducting informal international meetings where significant research in Artificial Intelligence can be presented and discussed. The 1986 conference continues this tradition. The call for papers resulted in the submission of over 100 manuscripts for the program committee and their referees. As a result we have been able to assemble a program of top quality research papers. Many good papers were rejected in order to keep the technical program manageable in size and to preserve the informal nature of the conference. Indeed, we have tried to schedule sufficient time outside the formal paper presentations for participants to meet and discuss their work.

I would like to take this opportunity to recognize some of the people who made this technical program materialize. All their effort was volunteered and the conference would not have been possible otherwise. I want to thank the program committee who were ultimately responsible for the review process. They cheerfully handled an overload of papers without forewarning of the extra workload involved. As well, the deluge of contributed papers forced me to conscript additional program committee members at the last minute. To these individuals, thank you for your willingness to serve on such short notice. The referees who read the submitted papers also deserve our gratitude. Critically judging others work is itself hard work involving a great deal of responsibility. Thank you all for your efforts promptly contributed. Finally, we thank those people who submitted their efforts to the less visible but essential tasks of preparing the call for papers, designing the advertising, receiving and distributing manuscripts, corresponding with authors and the program committee, and performing other unrecognized feats of necessity. They include Kathy Finter, Nadine Lasalle, Sandy Sapers, May Vink, and Lindsey Wey.

In the remainder of these proceedings you will find the results of our labours. It is our hope that the research reported here will make a significant contribution to progress in the discipline of Artificial Intelligence. Please enjoy !

Bill Havens

Program Chairman

EXÉCUTIFS DE LA SCEIO / CSCSI OFFICERS

1984 - 1986

PRÉSIDENT :

D[r] Gordon McCalla
Department of Computational Science
University of Saskatchewan
Saskatoon (Saskatchewan)

VICE-PRÉSIDENT :

D[r] John Tsotsos
Department of Computer Science
University of Toronto
Toronto (Ontario)

SECRÉTAIRE:

D[r] Michale Bauer
Department of Computer Science
University of Western Ontario
London (Ontario)

TRÉSORIER :

D[r] Wayne A. Davis
Department of Computing Science
University of Alberta
Edmonton (Alberta)

COMITÉ DU PROGRAMME - SCEIO 1986

PROGRAM COMMITTEE - CSCSI 1986

PRÉSIDENT :

Dr William Havens
Dept. of Computer Science
University of British Columbia

Dr Roger Browse
Dept of Computer Science
Queens University

Prof. Nick Cercone
School of Computing Science
Simon Fraser University

Dr B. Chandrasekaran
Dept. of Computer Science
Ohio State University

Dr Erni Chang
Alberta Research Council

Prof. Ted Elcock
Computer Science
University of Western Ontario

Dr Brian Funt
Dept of Computing Science
Simon Fraser University

Dr Jay Glicksman
AIDS
Mountain View, (CA)

Dr Randy Goebel
Dept of Computer Science
University of Waterloo

Dr Geoff Hinton
Computer Science Department
Carnegie-Mellon University

Dr Graeme Hirst
Dept of Computer Science
University of Toronto

Dr Hector Levesque
Dept of Computer Science
University of Toronto

Prof. Alan Mackworth
Dept of Computer Science
University of British Columbia

Prof. Gordon McCalla
Dept of Computational Science
University of Saskatchewan

Dr Robert Mercer
Dept of Computer Science
University of Western Ontario

Dr Jan Mulder
Dept of Statistics, Math & Computer Sc.
Dalhousie University

Dr Richard Rosenberg
Dept of Computer Science
Dalhousie University

Dr Len Schubert
Dept of Computer Science
University of Alberta

Dr Robert Woodham
Dept of Computer Science
University of British Columbia

ARBITRES / REFEREES

ADOLPH, Steve
ARMSTRONG, Bill
BAILES, Alison
BARRIE, James
BAUER, Michael
BOYER, Michel
BRATLEY, Paul
BRUNSON-LOKER, Barbara
BUTLER, Brian
BUTLER, Jean
CHVATAL, V.
COHEN, Robin
DAHL, Veronica
DAWES, Roger
DELGRANDE, Jim
EAGER, Derek
ELIO, Renee
GAMBLE, Ken
GLASGOW, Janice
HADLEY, Bob
HAKEN, Armin
HARRISON, Phil
HAYWARD, Vincent
HEBERT, Martial
JAGANNATHAN, V.
JENKINS, Mike
KASKERSKI, R.
KRAMER, Bryan
KUSALIK, Tony
LESPERANCE, Yves
LINSKY, Bernard
MAJUMDAR, Shikharesh
MARLEY, A.A.
MASRANI, Roy
MAXWELL, Michael
MC CROSKY, Carl
MERRETT, Tim
MEWHORT, Doug
MURPHY, Arthur
NOWLAN, Steven
PEARLMUTTER, Barek
PELLETIER, Jeff
PLANT, David
PLANTINGA, Ed
PLANTINGA, Edwin
POOLE, David
PRINTZ, Henry
ROWAT, Peter
STRZALKOWSKI, Tomek
SUTHERLAND, Lynn
SZELISKI, Richard
TOUREKZKY, David
TSOTSOS, John
van EMDEN, Martin
VAUCHER, Jean
WASSON, Barbara
WHITESIDES, Sue
XIE, Shun-En
YANG, Herb

COMITÉ ORGANISATEUR - SCEIO 86
ORGANIZING COMMITTEE - CSCSI 86

PRÉSIDENT :

Dr Renato De Mori
School of Computer Science
UNIVERSITÉ MC GILL

TRÉSORIER :

Dr Lorne H. Bouchard
Dépt de math et d'informatique
UNIVERSITÉ DU QUÉBEC À MONTRÉAL

SECRÉTAIRE :

Mlle Nadine Lasalle
CENTRE DE RECHERCHE INFORMATIQUE
DE MONTRÉAL INC.

Dr Daniel Delmas
Dépt de génie mécanique
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Dr Evelyne Hausen-Tropper
Dépt de math et d'informatique
UNIVERSITÉ DU QUÉBEC À MONTRÉAL

Dr Rajjan Shinghal
Dept of Computer Science
UNIVERSITÉ CONCORDIA

Dr Jean Vaucher
Dépt d'I.R.O.
UNIVERSITÉ DE MONTRÉAL

Dr Steven Zucker
Dept of Electrical Engineering
UNIVERSITÉ MC GILL

CONFÉRENCIERS INVITÉS
INVITED SPEAKERS

"Why Kids Should Learn to Program"

D[r] Eliot Soloway / Yale University

"Modeling Discourse Structure: The Role of Purpose in Discours"

D[r] Candy Sidner / Bolt Beranek & Newman Inc.

"How Much Thinking is Deduction ?"

D[r] Patrick J. Hayes / Schlumberger Laboratory, Palo Alto

"A Vision System for Autonomous Navigation"

D[r] Takeo Kanade - Robotics Institute, Carnegie-Mellon University

"The Role of Constraints in Problem Solving"

D[r] Mark S. Fox - Robotics Institute, Carnegie-Mellon University

"The Correspondence Continuum"

D[r] Brian C. Smith - Xerox Palo Alto Research Center

TABLE DES MATIERES / TABLE OF CONTENTS

APPRENTISSAGE / LEARNING

RAISONNEMENT FORMEL / FORMAL REASONING

COMPRÉHENSION DES LANGAGES NATURELS
NATURAL LANGUAGE UNDERSTANDING

PROGRAMMATION LOGIQUE / LOGIC PROGRAMMING

RECONNAISSANCE ET PERCEPTION / RECOGNITION AND PERCEPTION

VISION / COMPUTER VISION

SYSTEMES EXPERTS / EXPERT SYSTEMS

REPRÉSENTATION DES CONNAISSANCES KNOWLEDGE REPRESENTATION

APPLICATIONS / APPLICATION METHODOLOGY

Why Kids Should Learn to Program

Elliot Soloway

Cognition and Programming Project
Department of Computer Science
Yale University

ABSTRACT — In the past, children were exhorted to learn Latin in school; the folklore had it that (1) since this language was the basis of all other Western languages, then it would help students in their language studies, and more generally, that (2) in learning Latin one developed a discipline of thinking that was useful not only in other intellectual pursuits, but in everyday life as well. Today, the same exhortation is made for learning to program a computer — and the same sorts of myths are used to support the importance of learning to program. Unfortunately, there is precious little empirical evidence that supports the claim that learning to program imparts any more knowledge than simply learning to program. Nonetheless, even without solid empirical support, there is still the widespread belief that learning to program *does* provide kids with *something* more than just programming skills, i.e., that the skills learned in programming do transfer to other problem solving contexts. While it wouldn't be the first time that a commonly held belief turned out to be unjustified, the strength of those holding this conviction and the importance to education if they are right, suggests that we shouldn't give up the search for empirical support of transfer. In particular, in this paper we attempt to confront the transfer issue head on; we first briefly describe our efforts at finding empirical support for transfer, and we go on to suggest how the teaching of programming should be augmented in order to facilitate that elusive transfer.

RESUME — Jadis, on poussait les enfants à apprendre le latin à l'école; on prétendait que (1) puisque ce langage constitue la racine du reste des languages occidentaux, il aiderait les élèves dans leurs études linguistiques, et plus généralement, que (2) apprendre le latin développait une discipline du raisonnement utile non seulement (non solum) dans d'autres domaines intellectuels, mais également (sed etiam) dans la vie quotienne. Aujourd'hui, les mêmes exhortations sont faites en ce qui concerne l'apprentissage de la programmation, et des mythes du même ordre sont utilisés pour en étayer l'importance. Malheureusement, il y a bien peu de preuves empiriques qui suggèrent qu'apprendre à programmer apporte une connaissance qui aille au-delà du simple "savoir programmer." Néanmoins, malgré l'abscence de validation empirique, une croyance largement répandue subsiste qui veut qu'apprendre à programmer confère aux enfants quelque chose de plus qu'une simple maîtrise de la programmation, i.e., que les techniques acquises lors de l'apprentissage de la programmation soient effectivement transférables à d'autres contextes de résolution de problèmes. Bien que celà ne serait pas la première fois qu'une croyance communément répandue s'avérerait injustifiée, la vigueur de ceux qui adhèrent à cette conviction et l'importance de leur jugement, si toutefois ils ont raison, vis à vis de l'éducation suggèrent qu'il serait mal avisé de renoncer à rechercher des preuves empiriques d'un tel transfert d'expertise. Dans le présent article, nous tentons d'adresser directement ce problème de transfert: en premier lieu, nous décrivons brièvement les efforts que nous avons menés pour rechercher des preuves empiriques de transfert, puis nous suggérons comment l'enseignement de la programmation devrait être étendu de façon à faciliter cet insaisissable transfert.

1 Introduction: Motivation and Goals

In the past, children were exhorted to learn Latin in school; the folklore had it that (1) since this language was the basis of all other Western languages, then it would help students in their language studies, and more generally, that (2) in learning Latin one developed a discipline of thinking that was useful not only in other intellectual pursuits, but in everyday life as well. Today, the same exhortation is made for learning to program a computer — and the same sorts of myths are used to support the importance of learning to program. Unfortunately, there is precious little empirical evidence that supports the claim that learning to program imparts any more knowledge than simply learning to program. Nonetheless, even without solid empirical support, there is still the widespread belief that learning to program *does* provide kids with *something* more than just

This work was sponsored by the National Science Foundation, under NSF Grants MDR-8470150 and DPE-8470014.

programming skills, i.e., that the skills learned in programming do transfer to other problem solving contexts. While it wouldn't be the first time that a commonly held belief turned out to be unjustified, the strength of those holding this conviction and the importance to education if they are right, suggests that we shouldn't give up the search for empirical support of transfer. In particular, in this paper we attempt to confront the transfer issue head on; we first briefly describe our efforts at finding empirical support for transfer, and we go on to suggest how the teaching of programming should be augmented in order to facilitate that elusive transfer.

The organization of this paper is as follows: in Section 2 we highlight our efforts at exploring transfer effects from programming to algebra, i.e., does learning to program help students perform more effectively on algebra word problems? In the face of the mixed results we have obtained in that effort, we step back, and in Section 3 suggest what *should* be taught in programming classes — and argue why that new curriculum might provide the needed leverage on facilitating transfer.

2 The Relationship of Programming to Algebra: A Brief Synopsis

In what follows, we present our research on transfer effects, from programming to algebra, in chronological order, in order to convey a sense of the basis for and evolution of our thinking on transfer effects.

2.1 The Students and Professors Problem

Consider the following word problem:

> Given the following statement: 'There are six times as many students as professors at his University.' Write an equation to represent the above statement. Use S for the number of students and P for the number of professors.

Clement, Lochhead and Monk [2] found that only 63% of the college freshman engineering students were able to answer this problem correctly. Moreover, they found that only 27% of those students could answer the following problem correctly:

> Given the following statement: 'At Mindy's restaurant, for every four people who order cheesecake, there are five people who order strudel.' Write an equation to represent the above statement. Use C for the number of cheesecakes and S for the number of strudels ordered.

These problems appear to be simple algebra word problems, ratio problems, that students should have been able to solve by the time they were in college — and enrolled in an engineering program! Clement and Lochhead expended considerable energy establishing the robustness of the effect (e.g., wording doesn't appear to be the stumbling block: students perform the same when given pictures of objects to work from).

Insight into the source of the students' misconceptions comes from observing that the most typical incorrect answer was `6S = P`, instead of `S = 6P` for the first problem and `4C = 5S`, instead of `4S = 5C`, for the second problem. It appears that an answer such as `6S = P` is a *description* of a situation as opposed to a *prescription* for action. In video-taped interviews with subjects doing similar ratio problems, Clement and Lochhead found subjects who would make statements supportive of this 'declarative' interpretation.

2.2 The Contribution of Programming

At this point, I joined Clement and Lochhead's group and we hit upon the following hypothesis: if students are perceiving algebra to be a declarative description, and if we could put students in an environment that would encourage them to view the algebra in a more procedural, active manner, then maybe students would be able to solve these ratio problems more effectively. A prime candidate for such an environment is, of course, programming. This position, that programming encourages one to take an active, procedural view, is one that had been espoused by Papert [8] for some time.

Motivated by the above hypothesis we carried out a number of studies in various programming classes ([14,3]). For example, half the students in a programming class would be given a ratio problem (or problems) such as the PROGRAM VERSION in Figure 1, while the other half would be given a ratio problem such as the ALGEBRA VERSION in Figure 1. Given the type of results depicted in Figure 1, it appears that our hypothesis had some validity: students who were asked to write a program were more often correct than were students who were asked to write an algebraic equation.

Since the actual algebraic equation is the same in both situations, we felt that we had indeed tapped into something about programming that was the key contributing factor to the students' success. Our sense was that students in programming developed a very different notion of what a variable is, when compared with the notion of variable that students developed in the more traditional algebra context. That is, a variable in a program contains values that get acted upon by operators. In contrast, students who wrote down an answer such as `6S = P` may be viewing the `S` as a *label*, similar to 6feet = 1yard.

Based on the above success, we were encouraged to form the following hypothesis: if programming helps students to develop a more accurate view of what a variable is, then maybe this view can transfer back to algebra: maybe after taking programming, students can solve algebra problems more effectively? We are about to run a study in which we

THE PROGRAM VERSION

At the last company cocktail party, for every 6 people who drank hard liquor, there were 11 people who drank beer. Write a computer program in BASIC which will output the number of beer drinkers when supplied (via user input at the terminal) with the number of hard liquor drinkers. Use H for the number of people who drank hard liquor, and B for the number of people who drank beer.

THE ALGEBRA VERSION

At the last company cocktail party, for every 6 people who drank hard liquor, there were 11 people who drank beer. Write an equation that represents the above statement. Use H for the number of people who drank hard liquor, and B for the number of people who drank beer.

	Sample Size	% Correct	% Incorrect
PROGRAM VERSION	52	69	31
ALGEBRA VERSION	51	45	55

Probability of these results on the assumption that errors on problem are equally likely is $p < .05$.

Figure 1: Sample Results: Programming vs. Algebra

compare the performance, on *algebra problems*, of students in a programming class with students who have not had programming. We will use the standard pre- and post-test paradigm; if transfer does occur then we expect that the students who had programming will do much better on the post-test than the students who did not have programming and that both groups will perform the same on the pre-test. Pilot studies along these lines have been carried out with mixed results. Moreover, as we argue in the next section, we may be like the drunk who looks for his keys under the lamppost where the light happens to be, not in the place where he has lost them.

2.3 Status Report: In Search of the Elusive Transfer Result

Based on the observation made in Section 2.2 that students were able to write programs more effectively than algebraic equations, it seems quite reasonable then to focus on the potential specific factors involved: the differences between the notion of variable in programming and in algebra, the differences in the notion of the equal sign, etc. However, the mixed results from our pilot studies in looking at transfer to algebra call this rationale into question. While we clearly must await the results of the actual study and only use the pilot studies as a guide, we are nonetheless plagued by the following concern: maybe we are focusing on too literal a relationship between algebra and programming, e.g., maybe the link via the notion of a 'variable' is too narrow a view.

That is, consider what it would mean for students who took the programming course to perform better on the algebra post-test because of their new improved notion of a variable:

- The notion of variable that they learned in programming simply 'overwrote' the old notion of variable that they had learned in algebra. Thus, when students came to solving problems in algebra, after they had taken a programming course, they simply accessed this new, programming notion of a variable. Frankly, this account seems implausible: why should the notion of 'variable' learned in algebra be wiped out when learning the notion of 'variable' in a new domain; it would seem more likely that a new notion of 'variable' would be created *in the context of programming* and that the old notion of variable, learned in the context of algebra, would simply remain.

- However, if two notions of variable existed, each indexed by the context in which it was learned (programming and algebra) then why would we expect to see students using the new, programming notion of variable when trying to solve algebra problems? Wouldn't it be more reasonable to expect subjects to retrieve the notion of variable learned in the algebra context since they were attempting to solve algebra problems?

Thus, it is not clear that we should even expect the students in the programming class to do better on the algebra post-test; while they may indeed possess a more effective notion of variable, learned in the context of programming, it does not seem clear that they will even be able to access it when solving the algebra problems. However, if we don't find transfer to algebra, then we must be surprised: surely, learning to program must transfer to such a close domain as algebra. On the other hand, given the sort of arguments made above, it is not clear that there should be transfer even at this low a level. It is arguments such as the above that keep us awake at night!

In the next section, we will argue that we might want to look for transfer at a level much higher than variables, etc. In particular, we attempt to show that students learning programming, when the programming curriculum is redefined, might be able to learn more general skills that may in fact transfer to other domains.

3 Revising The Programming Curriculum To Facilitate Transfer

3.1 Teaching Generic Tasks: Constructing Mechanisms and Explanations

By and large, introductory programming textbooks stress the syntax and semantics of a programming language.[a] Some textbooks attempt to teach problem solving in the context of teaching programming; while well-intentioned, these books simply don't typically say enough explicit about problem solving. For example, top-down, step-wise refinement is like a catechism for teaching good problem solving/programming habits. Almost all texts attempt to teach this technique. However, they typically give examples of the use of the technique — not how a novice should learn to carry out that technique. While most instructors of introductory programming courses would shun the idea that they are simply teaching a programming language, the fact is that the tools they use - the textbooks — don't help to impart good problem solving concepts. The bottom line is this: if we are looking for transfer effects, we will be hard put to find them in courses that simply attempt to teach the syntax and semantics of a particular programming language. Why should learning where the semi-colon must be placed in a Pascal program have any impact on learning geography?

A closer look at what a program is meant to accomplish provides the starting point for our view of what should be taught in programming. In particular, we see a program as having two major audiences, a computer and a human reader:

- to the computer, the program provides a *mechanism* for *how* to solve the problem;
- to a human reader, the program provides a justification or *explanation* as to *why* the problem is solved by the program.

In their commonsense usage, a 'mechanism' is simply the means to achieving some goal(s), while an 'explanation' provides the reasons for the actions ([10]). In the case of programming, the actions are the statements of the program, the reasons are the goals being achieved, and the plans are the standard manner in which to achieve those goals.

For example, in the right half of Figure 2, a Pascal program is depicted that will satisfy the problem statement given at the top of that figure and output the average of the numbers input. The program is the mechanism that instructs the computer as to how an average should be calculated. Thus:

> *How* does the program in Figure 2 solve the desired problem? Well, when executed by a computer (or by a human playing computer), the program accepts numbers from a user, adds those numbers into an accumulator called `total`, etc.

Note that the description given in Figure 2 is only a partial representation of the mechanism. That is, in executing the mechanism, the computer turns a static program, written in two dimensions on a piece of paper, into a the dynamic, three-dimensional entity that exists over time. In this dynamic representation notions such as the causal relationship between the statements become very important and must be used in describing how a program works: e.g., the running total (**`total`**) and counter update (**`count`**) is done before the next input value is read in; after the sentinel value is input, then a test is made to see if the counter is greater than zero.

In the left half of Figure 2, we depict an explanation for why that program computes the desired answer. The goals of the English statement of the problem are identified and stereotypic methods (plans) are used to attack and resolve each of these goals ([15,11]). Thus:

> *Why* does the program effectively compute the average of the numbers read in? Well, the SENTINEL CONTROLLED LOOP RUNNING TOTAL LOOP PLAN is the appropriate plan for achieving one of the subgoals of the problem, namely: 'repeatedly reading in and summing data until a final stopping value is input.' This plan is appropriate because the problem states that the loop is terminated following input of a specific value (the 'sentinel controlled' part of the plan), and because a running total is the appropriate plan for achieving the goal of summing the numbers input. Also, we need to merge a COUNTER LOOP PLAN with this SENTINEL CONTROLLED LOOP RUNNING TOTAL LOOP PLAN in order to keep track of the number of numbers that are input. Etc.

Just as the program description does not convey all there is to the mechanism, the goal/plan representation depicted in Figure 2 leaves off important pieces of information that would be included in a more complete explanation. For example, one might ask: how do you know that the merging of the COUNTER LOOP PLAN with the SENTINEL CONTROLLED RUNNING TOTAL LOOP PLAN will be correct? Or, why didn't you choose to read the numbers into an array, using a SENTINEL CONTROLLED ARRAY INPUT LOOPING PLAN and then process the data? Answers to these questions 'hang off' of the various concepts in the goal/plan representation.

[a] There are some introductory texts that attempt to teach the more technical aspects of computer science, e.g., properties of algorithms. However, typically these texts still don't teach the topics that will be discussed in this section either.

Problem: Write a program that will repeatedly read in and sum data until a final stopping value of 99999 is input. Output the average of the numbers read in.

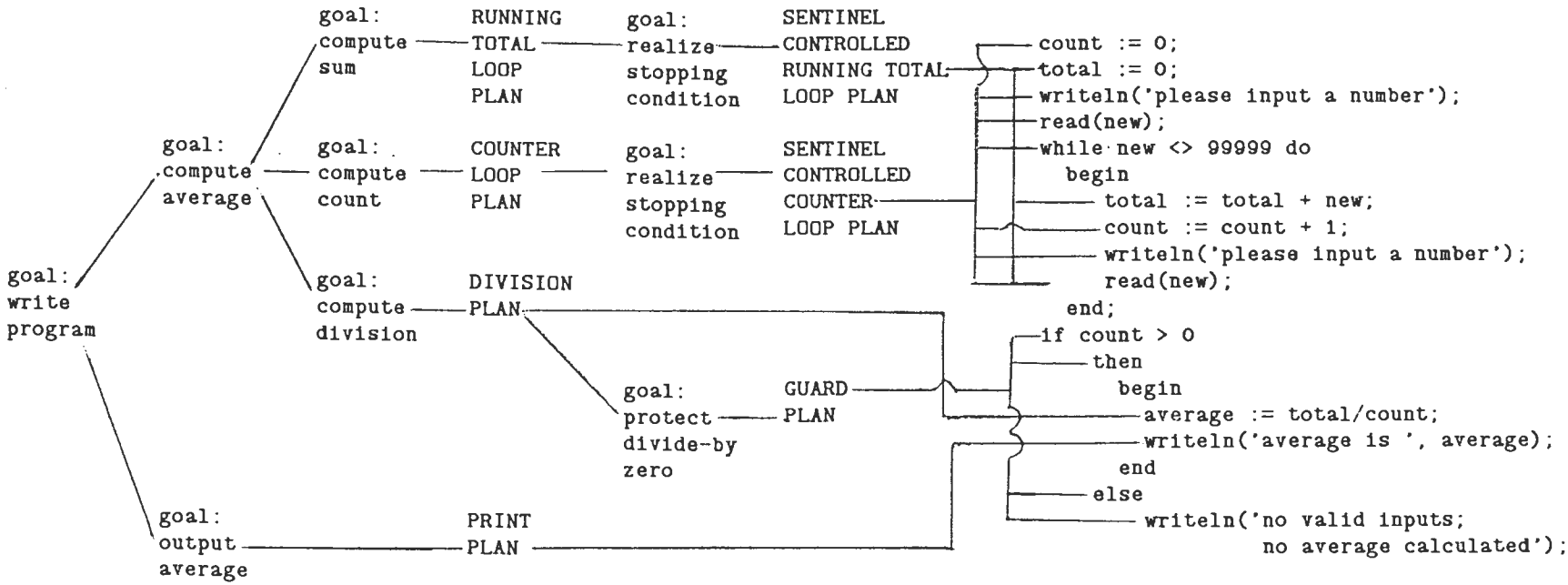


Figure 2: An Example: Mechanism and Explanation

It is our claim, then, that leverage on the transfer issue can come from viewing learning to program as really learning to perform two generic tasks: *construct an explanation* and *construct a mechanism*. These two tasks are generic in the sense that people must perform them all the time in their everyday lives. That is, the need to explain why something works or the need to develop a set of procedures for accomplishing a goal is not limited to computer programming; the ability to carry out such tasks is constantly and naturally being demanded by the very fact that people interact with other people and with artifacts created by people. The goal, then, in our revised view of what the programming curriculum should be, is to teach the skills needed to construct mechanisms and explanations. (Section 3.2 identifies some of these skills.) Moreover, instruction in programming must make explicit the fact that what is being taught are skills for constructing explanations and mechanisms; without this explicitness, students may not realize that they are learning skills for generic tasks.

If constructing explanations and mechanisms is so ubiquitous, why should they be taught in the context of programming? Why can't they be taught in many other contexts equally as effectively? In fact, we think that the skills to enable people to carry out these tasks are taught in most all other disciplines; however, it is our reasoned belief that, in contrast to such domains as physics or geometry, programming provides an exceedingly effective environment in which to teach people such skills. For example, in teaching physics, one is certainly exposed to mechanisms. However, in physics the notion of teleological explanation is not one that is favored: why is it the case that when one puts equal weights on each side of a balance beam equidistant from the fulcrum, that the balance beam remains horizontal? Or, why does the balance beam remain horizontal, when, on one side of the balance beam, one puts twice as much weight half as far away as the weight on the other side of the beam? There are no goals to appeal to here. In contrast, there are certainly explanations in a domain such as geometry: one needs to explain why angle-side-angle, a standard plan in geometry, works in a particular case. However, domains such as geometry are so technical, so specialized, that transfer of the skills needed to construct such explanations is difficult; quite reasonably, it is hard to see the relationship between angle-side-angle and deposit-more-money-in-the-checking-account, a plan is that is used to solve all sorts of problems in the everyday world.

In contrast, then, to domains such as physics and geometry, programming allows one to be explicit about explanations and mechanisms, and to teach those concepts in the context of everyday problems. That is, non-technical, non-science oriented classes are taught using a variety of problems, e.g., balancing and reconciling bank statements, projecting a course plan and potential grades. These sorts of problems are precisely the kind that students face all the time. Thus, one would expect that transfer out of programming would be facilitated, since students would be learning the skills of explanation and mechanism construction in familiar contexts.

There is, in fact, some intriguing empirical evidence that supports our claim that programming does teach students the skills necessary for constructing explanations. Howe, et al. [4] report a study in which a class of junior high school math students was split in half, with only one group receiving training in LOGO. On post-tests, Howe et al. found no differences in performance: both groups scored about the same with respect to correctness. However, teachers of the students noted that those students who had the LOGO training were much more able to explain their confusions in math than were the students without exposure to LOGO programming. While this intriguing observation must be replicated and quantified, it is still suggestive of the point we are trying to make: the benefit of programming lies in teaching the students to carry out two generic tasks – constructing explanations and mechanisms.

3.2 Skills Needed to Construct Explanations and Mechanisms

In this section we will attempt to identify three key skills that are involved in learning to effectively construct mechanisms and explanations: the use of (1) step-wise refinement, (2) plan composition methods, and (3) rules of programming discourse.

3.2.1 Step-Wise Refinement Essentially all textbooks on programming attempt to each students how to carry out step-wise refinement (top-down design). Step-wise refinement is a planning technique that is intended to provide an orderly method for thinking about a problem.[b] Typically the student is told that he/she should:

> Break a problem down into subproblems.

He/she is then shown how the author carries out this prescription on a set of examples. Students typically can follow the text's argument quite well: they can understand the transition between each step. However, when asked to create a step-wise refinement for a new problem, by and large students are at a loss as to where to even begin. When they try to carry out a step-wise refinement for themselves — and thus when they must confront the question of how to do a step-wise refinement — troublesome issues such as the following arise: (1) why was the problem broken down into those n levels; why not m levels? (2) and why were those particular routines used; why not some other routines? Texts typically don't answer those questions. Effectively, the students are left to induce the strategies for

[b] In fact, in our studies of professional program designers we have found that the experts do use step-wise refinement ([1]).

doing step-wise refinement on the basis of the textbook's examples.[c]

In teaching step-wise refinement, we add one crucial heuristic that everyone intuitively knows, but seldom makes explicit:

> Break a problem down into subproblems, *on the basis of problems that you have already solved and for which you have canned (or almost canned) solutions.*

In teaching step-wise refinement, the image we try to paint is as follows:

> Assume that you already possess a 'barrel of canned solutions.' Look up at the problem from that barrel of solutions, and try to see if some of those canned solutions can be used in the solution of the new problem. *Break the new problem down so that you can use those canned solutions.*

In otherwords, one must already possess the primitives into which the problem will be decomposed in order to follow a step-wise refinement strategy.

Canned solutions are what we have been calling programming plans — stereotypic methods for achieving goals. Thus, the objective of the step-wise refinement planning strategy is to identify in the given problem statement various goals for which one has already developed programming plans. What if one doesn't have the plans that are relevant to a problem? For example, what happens when students just start learning to program? The answer, frankly, is quite simple: one can't employ step-wise refinement as straightforwardly; one will do a substantial amount of floundering and searching, and one will tend to decompose a problem inappropriately. In fact, in our study with junior grade software designers this floundering behavior is precisely what we observed ([1]). If one doesn't have a barrel of canned solutions, then one can't apply a step-wise refinement strategy.

The process of generating a goal/plan decomposition for a given problem via step-wise refinement produces an explanation for why the resultant mechanism is an effective solution to the problem: the goal/plan decomposition relates goals that need accomplishing to techniques (plans) for achieving those goals. Moreover, as part of the goal/plan decomposition, the reasons why one goal and/or plan was chosen over another should also be made explicit; this too becomes part of the explanation.

3.2.2 Plan Composition Methods Step-wise refinement describes the 'macro strategy' for developing a goal/plan decomposition for a given problem. However, we have also identified rules for how one should weave together the plans at the 'micro level.' For example, in solving the Averaging Problem given in Figure 2, using step-wise refinement as we have defined it, one can see the need for an output goal, to print out the computed value, and a calculate goal, to produce the average. How should the plans which realize these two goals be 'glued together?' Since the CALCULATION PLAN is completely finished before the OUTPUT PLAN can be performed, one can simply *abut* the two plans — the plan for computing the average outputs a value which is the input to the plan that prints out the value.

More generally, we have identified 4 strategies for gluing together plans:

- *Abutment:* Two plans are glued together back to front, in sequence, as illustrated in the above paragraph.
- *Nesting:* In this strategy, one plan is completely surrounded by another plan. For example, in the Averaging Program given in Figure 2, we can see that the OUTPUT PLAN, the plan which realizes the goal of writing out the average, is nested within a SKIP GUARD PLAN, which realizes the goal of not permitting a divide by zero to occur in the average calculation if no numbers are actually input.
- *Merging:* In this strategy, two plans are interleaved. For example, in the Averaging Program given in Figure 2, we can see that two looping plans are merged to create a looping plan that will keep both a running total and a counter update. In particular, a SENTINEL-CONTROLLED RUNNING-TOTAL LOOP PLAN must be interleaved with a SENTINEL-CONTROLLED COUNTER LOOP PLAN in order to achieve the goals of keeping track of the sum and keeping track of the number of numbers summed, respectively.
- *Tailoring:* Sometimes a plan that one has already developed is not quite what is needed in a problem. One then needs to modify it so as to fit the particular needs of the situation. Afterall, we do call it 'software.'

Plan merging is an exceedingly difficult activity to carry out effectively. In studying bugs that novice programmers make, we see that when novices try to merge plans they almost invariably do it incorrectly [15]. Such behavior is not really all that surprising: weaving together two or more plans requires great care and attention to details, and the ability to foresee all sorts of subtle interactions. Thus, from an instructional point of view, we need to alert students to the problems of producing programs in which plans are merged.

In sum, then, by providing students with 4 strategies for gluing plans together, we are attempting to make explicit

[c] Anderson has seen a similar problem when students are taught how to develop proofs in geometry: students can usually follow the teacher's proof as he/she goes over it on the blackboard, but are stymied when faced with creating their own proofs while doing their homework ([9]).

```
Program Average
VAR count : INTEGER;
    sum, average, number : REAL;
BEGIN
 sum := -99999;
 count := -1;
 REPEAT
       writeln('please input a number');
       read (number)
       sum := sum + number;
       count := count + 1;
    UNTIL (number = 99999);
 average := sum/count;
 writeln('the average is: ',average);
END.
```

Figure 3: Violating A Rule of Programming Discourse

a strategy for developing the explanation and mechanism components of a program at the micro-level. In particular, (1) the explanation is furthered when one is told how the subgoals are being stichted together in order to solve the overall goal, and (2) the mechanism is furthered when the plans become instantiated in terms of actual programming language constructs.

3.2.3 Rules of Programming Discourse When plans are instantiated via actual code, there is still a great deal of flexibility left in as to how those plans should be realized: different language constructs can be used to realize the same plan and/or the statements in a plan can have multiple orderings. For example, consider the program in Figure 3, which does calculate and output a correct average. However, a teacher of programming would certainly not want to give the writer of this program full credit, since this program is written in a 'poor style:' initializing variables to some strange value (i.e., a value other than 0 or 1) is, generally speaking, not a good habit. A goal/plan analysis of this program reveals the students underlying intentions in performing these strange initializations:

> In attempting to realize the goal of reading in, summing and counting, user input integers, the student has implemented the SENTINEL-CONTROLLED RUNNING-TOTAL LOOP PLAN with a **repeat** construct in Pascal. However, the **repeat** construct is not appropriate when implementing a loop in which the stopping value can be input on the first read; in such a case the processing loop must not be executed even once. In Pascal, the **while** construct is the most appropriate loop for a SENTINEL-CONTROLLED RUNNING-TOTAL LOOP PLAN. (See [13].) The result of using a **repeat** construct is to allow the sentinel value to be added inappropriately into the running-total (and also the counter is inappropriately updated). To compensate for this 'off by one bug' the student has subtracted out the sentinel value (and the counter value) during the initialization phase of the program!

Surely, backing out a value to compensate for an off by one bug is not a good programming practice.

Programming teachers, in marking down a student for writing a program such as the one depicted in Figure 3, often get the following response from students: 'But my program runs....' Moreover, oftentimes teachers are hard put to explicitly say why a piece of code is not in good style, and thus the student is often left in the dark as to why his program is marked down. Thus, besides teaching about step-wise refinement and plan composition methods, we also teach students to use *rules of programming discourse* ([5]), which are analogous to rules of conversational discourse. For example, consider the following interchange:

> Mary: Hi, John. How are you?
> John: Oh, the sky is blue.

In a normal context, John's response is inappropriate; following accepted rules of discourse, John should have said something like

> Fine, Mary, how are you?

Analogously, there are rules of programming discourse that prescribe good programming practices. For example, the program in Figure 3 violated the following rule of programming discourse:

> ***Don't do 'double duty' with code, especially when the second function played by the code is obscure.***

That is, the initialization of **Sum** and **Count** played two roles:

- the variables were initialized to some starting value (which is typically 0 or 1),
- and the starting values were chosen so as to compensate for an inappropriately constructed loop

Clearly, that second role was obscure! With the exception of such books as Ledgard et al. [7] and Kernighan and Plauger [6], programming textbooks typically do not explicitly teach students about the rules of programming discourse. Rather, students are left to pick them up from observing examples of programs. However, as programming teachers know, students tend not to learn what counts as a program written in a good style.

From the computer's perspective, a program that violates the rules of discourse but computes the desired value is as good as a program that does not violate the rules of programming discourse. However, a human reader is greatly aided in understanding a program when that program is written obeying the rules of discourse.[d] In effect,

[d] We have found that rules of programming discourse play a critical rule in aiding program comprehension: an expert programmer uses these rules of programming discourse, albeit implicitly, and as-

such a program enables the reader to reconstruct the explanation that the program writer developed in generating the mechanism.

4 Concluding Remarks

There are essentially 3 reasons why one might consider teaching children to program:

- *Point 1:* As a job skill; students leaving a programming course would be better equipped to land a good paying position.
- *Point 2:* As a tool to be used; students leaving a programming course may have acquired the skills necessary to use the computer as a tool in solving problems in other domains (e.g., data analysis).
- *Point 3:* As an exemplar of skills that are used in other areas; learning to program is analogous to acquiring the Rosetta Stone: the student can now move between different problem domains and still be an effective problem solver.

If programming really only achieves Point 1, then its importance to education has been blown way out of proportion: programming is a job skill on par with drafting or welding and thus has no real place in an academic setting in which all students are required to take a core curriculum of which programming is a par; but rather, programming should be relegated to a technical curriculum where only interested students need come. Moreover, if programming only achieves Point 2, then maybe its inclusion in the core curriculum might be justifiable, but there still is substantial room to question the wisdom of requiring students to take programming. Of course, what has got everyone excited is Point 3: people have this intuition that learning to programming is somehow an enlightening experience and provides a student with more than just a skill or the ability to use a tool. Unfortunately, there are only glimmers of empirical evidence for Point 3. Moreover, as we have tried to argue in this paper, by and large programming is currently taught in such a manner as to provide little hope of of facilitating transfer.

However, we have suggested that the programming curriculum be revised in order to more effectively facilitate transfer. In particular, our position is that programming can — and should — be viewed as teaching the generic tasks of constructing explanations and constructing mechanisms. Moreover, rather than focusing on teaching programming per se, our goal is to introduce a new set of terms that can provide the student with a perspective on programming that is different from what is currently taught. For example, we have been advocating that terms such as explanation, mechanism, goal, plan, composition methods, abutment, etc. should be the subject of instruction in a programming course, along with the usual discussion of the syntax and semantics of a particular programming language. Viewed in this light, learning to program might provide students with skills that do transfer to other contexts in which explanations and mechanisms are needed.

sumes that other programmers will also use these rules. Thus, when a programmer is given a program that violates the rules of discourse, he/she often finds such programs very difficult to comprehend. In fact, in a recent study, we were able to reduce the performance of advanced programmers to that of novice programmers by asking the advanced programmers to deal with programs that violated various rules of discourse [12].

Acknowledgement

I would like to thank a number of people who played important roles in this work: to Roger Schank, who, by continually insisting that teaching programming per se is 'not all that important,' challenged me to think more deeply about why I believe in teaching programming; to Ben Shneiderman, with whom I have had many, many thought provoking discussions about programming; and to Judah Schwartz, who encouraged me to step back a bit and who provided just the right amount of nudging. I would also like to thank the members of the Cognition and Programming Project here at Yale; they are carrying out the research on the front lines, which gives me the chance to sit back and ponder.

References

[1] B. Adelson and E. Soloway. The role of domain experience in software design. *IEEE Transaction on Software Engineering*, November 1985.

[2] J. Clement, J. Lockhead, and G. Monk. Translation difficulties in learning mathematics. *American Mathematical Monthly*, 88(4):26–40, 1980.

[3] K. Ehrlich, E. Soloway, and V. Abbott. Styles of thinking: from algebra word problems to programming via procedurality. In *Proceedings of the Cognitive Science Conference*, Cognitive Science Society, Univ. of Michigan, Mich., 1982.

[4] J.A.M. Howe, T. O'Shea, and J. Plane. *Teaching Mathematics Through Logo Programming.* Technical Report 115, University of Edinburgh, Artificial Intelligence, 1979.

[5] S. Joni and E. Soloway. But my program runs! discourse rules for novice programmers. *Journal of Educational Computing Research*, Winter 1985.

[6] B. Kernighan and P. Plauger. *The Elements of Style.* McGraw Hill Co., New York, 1978.

[7] H. Ledgard, J. Hueras, and P. Nagin. *Pascal with Style: Programming Proverbs.* Hayden Book Co.,

Rochele Park, N.J., 1979.

[8] S. Papert. *Mindstorms, Children, Computers and Powerful Ideas.* Basic Books, 1980.

[9] P. L. Pirolli and J. R. Anderson. The role of learning from examples in the acquisition of recursive programming skills. *Canadian Journal of Psychology*, 39(2):240–272, 1985.

[10] R. C. Schank. *Explanation: A First Pass.* Technical Report YaleU/CSD/RR #330, Dept. of Computer Science, Yale University, New Haven, Ct., 1984.

[11] E. Soloway. A tale of one little program. In Karen Duncan and Diana Harris, editors, *Proceedings of the IFIP/AFIPS WCCE 85 World Conference on Computers in Education*, pages 553–559, Norfolk, Virginia, 1985.

[12] E. Soloway and K. Ehrlich. Empirical studies of programming knowledge. *IEEE Transactions on Software Engineering*, SE-10(5):595–609, 1984.

[13] E. Soloway, K. Ehrlich, J. Bonar, and J. Greenspan. What do novices know about programming? In B. Shneiderman and A. Badre, editors, *Directions in Human-Computer Interactions*, Ablex, Inc., New York, 1982.

[14] E. Soloway, J. Lochhead, and J. Clement. *Does Computer Programming Enhance Problem Solving Ability? Some Positive Evidence on Algebra Word Problems*, pages 171–215. Academic Press, New York, NY, 1982b.

[15] J. Spohrer, E. Soloway, and E. Pope. *A Goal/Plan Analysis of Buggy Pascal Programs*, pages 163–207. Volume 1, Lawrence Erlbaum Associates, Inc., Hillsdale, N.J., 1985.

GENERATIVE STRUCTURE IN ENUMERATIVE LEARNING SYSTEMS

Robert C. Holte

Department of Electrical Engineering and Electronics
Brunel University, Uxbridge, England UB8 3PH

R. Michael Wharton

Department of Computer Science
York University, Downsview, Ontario M3J 1P3

Abstract

As part of an incremental learning task, a learning system must replace its current hypothesis when it is found to be unacceptable. The internal organization of the process which generates the new hypothesis imparts a structure on candidate space, called "generative structure". The information indicating that the current hypothesis is or is not acceptable usually indicates the acceptability or unacceptability of a large number of other candidates as well, and it is important for a learning system to make effective and complete use of this information. In the specific case of *a priori* information characterizing unacceptable candidates, we demonstrate that generative structure is a major determinant of the effectiveness with which an enumerative learning system is able to use this information.

1. Introduction

In an incremental learning task (see Figures 1, 2 and 3), the learning system produces a sequence of hypotheses: the system succeeds if this sequence converges, in some sense, to the "target concept" (*i.e.* the one which is to be learned). In general there may be a set of target concepts, called the "target set".

The information supplied to the learning system by the external assessor is crucial to its success. Even when it is couched in terms of the current hypothesis, this information is a partial characterization of the target set. As this information accumulates an increasingly accurate characterization of the target set emerges.

Systems differ in how effectively they use the information supplied by the performance assessor. This "effectiveness" is a measure of net reduction in computational effort, based on the computational effort required to make use of the information and the computational effort saved by using this information. A question which naturally arises is, "What aspects of a system determine the effectiveness with which it can use this information ?"

This paper analyzes the effect of one aspect of an enumerative system, its generative structure, on the effectiveness with which the system can exploit "*a priori* exclusion information", *i.e.* information given to the system before it has produced its first hypothesis, and which characterizes some of the candidates which are not in the target set.

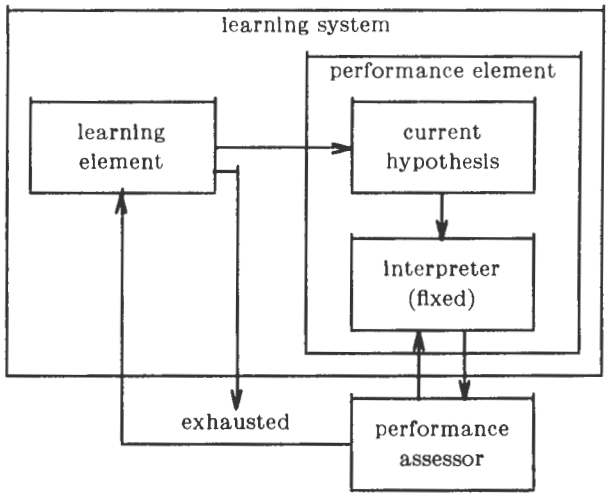


Figure 1. Incremental learning task. As in [1], an incremental learning system is modelled as consisting of a learning element and a performance element. The performance element is modelled as consisting of a hypothesis, supplied by the learning element, and a fixed interpreter. A performance assessor, outside the learning system, interacts with the performance element, and supplies the learning element with information comparing the performance resulting from the current hypothesis to a target performance. The line labelled "exhausted" is used to signal that the learning element is only able to produce candidates which are known to be unacceptable.

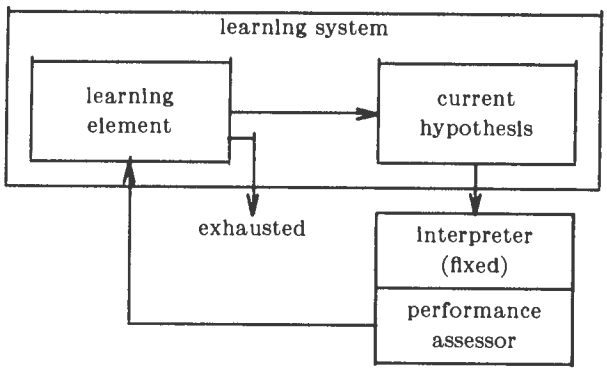


Figure 2. Incremental learning system - boundaries redrawn. The interpreter is combined with the performance assessor.

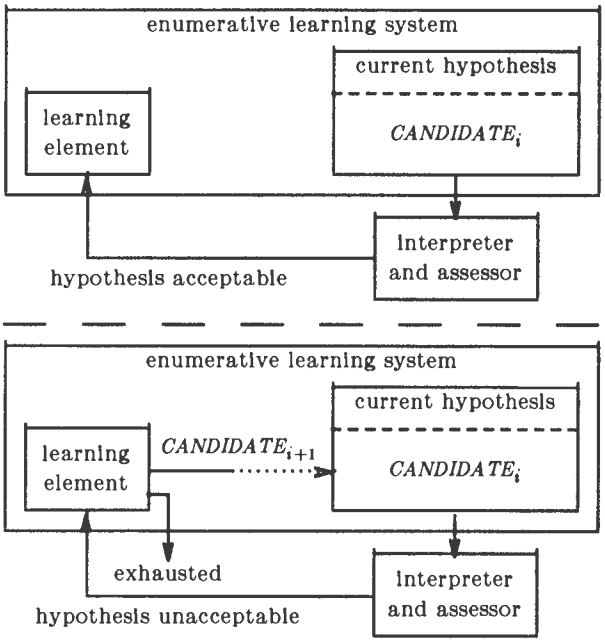


Figure 3. Enumerative learning system. (upper diagram) The current hypothesis, $CANDIDATE_i$, is retained as long as the information supplied by the assessor indicates that it is acceptable. (lower diagram) When the information supplied by the assessor indicates that $CANDIDATE_i$ is unacceptable, an enumerative learning system generates $CANDIDATE_{i+1}$, the next candidate in a predefined sequence, to replace $CANDIDATE_i$ as current hypothesis.

2. Generative Structure

A structure of a set S is a collection of sets $S_1, S_2, \cdots, S_k$ and an onto function F such that

$$F: S_1 \times S_2 \times \cdots \times S_k \rightarrow S$$

The S_i are not necessarily subsets of S. Each S_i is called an "immediate constituent" of S (with respect to F), and each S_i may itself have a structure. F is called a "composition function". A structure of S involving composition function F is called an "F-decomposition".

The three composition functions which are considered below are defined in terms of the standard binary operators set union (denoted $\cup$), difference (denoted $-$) and product (denoted $\times$) [a].

$$UNION(S_1, \cdots, S_k) = S_1 \cup S_2 \cup \cdots \cup S_k$$

$$DIFFERENCE(S_1, \cdots, S_k) = S_1 - S_2 - \cdots - S_k$$

$$PRODUCT(S_1, \cdots, S_k) = S_1 \times S_2 \times \cdots \times S_k$$

A "generator" for a set S is a procedure which produces a sequence of elements of S in which every element of S eventually occurs. For a given generator G of set S, structure $<\{S_1, S_2, \cdots, S_k\}; F>$ of S is defined to be the generative structure of S if G proceeds by generating each of the S_i and composing their elements as specified by F. G is said to F-decompose S into $S_1, \cdots, S_k$.

A generator union-decomposes set S into immediate constituents $S_1, S_2, \cdots, S_k$ if it proceeds by generating S_1, then S_2, and so on, and

$$S = UNION(S_1, \cdots, S_k)$$

The structure $<\{S_1, S_2, \cdots, S_k\}; UNION>$ is the generative structure of S associated with this generator. $S_1, \cdots, S_k$ are called union-constituents of S (see Figure 4).

Product-decomposition and product-constituents are defined similarly (see Figure 5). Note that product-constituents are not generally subsets of the original set S.

A generator difference-decomposes set S into immediate constituents $S_1, S_2, \cdots, S_k$ if it proceeds by generating the elements of S_1 and eliminating those which are elements of any of $S_2, \cdots, S_k$. This is the classic generate-and-test paradigm (see Figure 6), in which S_1 is the set which is generated internally and $S_2, \cdots, S_k$ are represented by the tests.

---

[a] The product of two sets, $P \times Q$, is normally the set of ordered pairs $<p,q>$ with p in P and q in Q. In this paper product is used to mean several slightly different mappings: the precise meaning is made clear from the examples which accompany the discussion. For instance, the product of two sets of strings, $U \times V$, is the set of all uv (concatenated) with u in U and v in V, and not the set of all $<u,v>$.

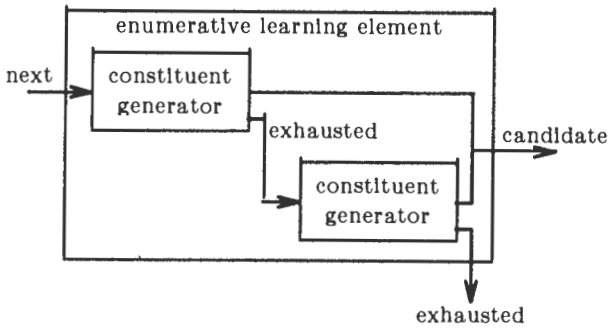


Figure 4. Union-decomposition generator with two sequential constituents.

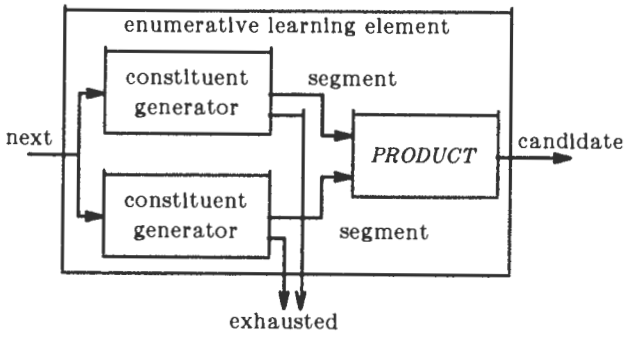


Figure 5. Product-decomposition generator with two constituents.

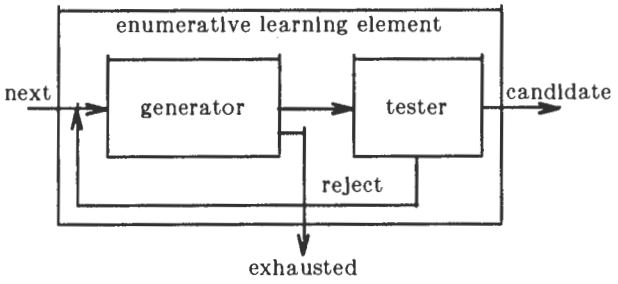


Figure 6. Difference-decomposition generator. Generate and test. The tester may contain several tests.

3. Reconfigurable Generators

Exclusion information characterizes some candidates, possibly very many, which are not in the target set. It is assumed that this characterization is stated in terms which can be easily computed for any given individual candidate. That is, it is straightforward to determine if a given candidate satisfies the characterization or not.

In an enumerative system for candidate set C, the problem of using *a priori* exclusion information characterizing a set E of candidates reduces to the problem of reconfiguring the system to generate the set $C-E$. The idea of a reconfigurable generator arises elsewhere in the artificial intelligence literature. For instance, the plan-generate-test paradigm on which DENDRAL [2] is based explicitly requires generators which can be reconfigured.

The most straightforward way to reconfigure the system to enumerate $C-E$ is to add the characterization of E as a test to be applied to the individual candidates produced by the system. Any characterization can be incorporated into any enumerative system in this way. However, this is the least effective way to use the characterization: the same number of candidates must be generated, and the computational cost of testing each candidate has been added.

The most effective way to use *a priori* exclusion information is to reconfigure the system so that it does not generate any of the candidates which have been characterized. In some circumstances, this kind of reconfiguration can be done easily. For instance, if the enumerative system union-decomposes candidate space, and the exclusion information happens to identify one of these union-constituents as containing no candidates in the target set, the enumerative system can simply skip over that union-constituent. In this way, the number of candidates produced is maximally reduced with little or no overhead.

Between these two extremes of effectiveness - testing individual candidates one by one after they have been fully generated, and preempting the generation of the entire set of characterized candidates - lies a range of intermediate possibilities which vary along two dimensions: how many separate applications of the characterization are required to eliminate all of the characterized candidates from the generation sequence, and how early in the generation process the characterization can be applied.

The significance of generative structure is that it determines which kinds of reconfiguration are feasible. It therefore determines a maximum possible "position" along both dimensions of effectiveness. For instance, if few of the characterized candidates occur in each structural constituent, then many separate applications of the characterization are needed. Also, a characterization cannot be applied until a point in the generation process at which sufficient information is available to evaluate the terms in which the characterization is defined: the time at which such information becomes available is dependent on the generative structure.

4. Example of Alternative Generative Structures

The influence of generative structure on the effectiveness with which *a priori* exclusion information can be used by an enumerative system may be demonstrated by comparing the reconfigurability of two generators for the same candidate space, but which have different generative structures. Two such generators for the set of context-free grammars are described in [3].

4.1 A Generator of Context Free Grammars

The first generator, GEN_1 (originally called the "inefficient enumerator"), union-decomposes the set of context free grammars into a collection of "size classes" (originally called "complexity classes"). Each size class is characterized by a triple $<n,m,p>$ of non-negative integers. As a union-constituent of the set of context free grammars, each size class is itself a set of context free grammars. A context free grammar is in size class $<n,m,p>$ if and only if it involves exactly n nonterminals and contains exactly p productions whose longest production's right-hand side is exactly m characters long.

Each size class $<n,m,p>$ is union-decomposed into a collection of "shape classes" (originally called "structure classes"). A shape class is characterized by an n-tuple of non-negative integers, $<q_1,q_2,\cdots,q_n>$ for which $\sum q_i = p$. Shape classes are the union-constituents of size classes and so they too are sets of context free grammars. For a fixed ordering of the nonterminals, $N_1,\cdots,N_n$, a grammar is in shape class $<q_1,\cdots,q_n>$ if and only if for each i between 1 and n, it contains exactly q_i productions whose left hand side is N_i.

GEN_1 product-decomposes each shape class. This is best described in terms of the syntactic structure it imposes on the grammars in the shape class. A grammar is product-decomposed into two immediate constituents, a left hand side p-tuple and a right hand side p-tuple. The left hand side p-tuple is an ordered set of cardinality p containing the left hand sides of the productions in the grammar. The right hand side p-tuple is an ordered set of cardinality p containing the right hand sides of the productions. The productions in the grammar can be constructed from these two p-tuples simply by pairing each element in the left hand side p-tuple with the corresponding element in the right hand side p-tuple. For instance, GEN_1 product-decomposes the following grammar,

$GRAMMAR_1$:

$$X \rightarrow A \mid B$$
$$A \rightarrow bb \mid bbA$$
$$B \rightarrow bbb \mid bbbB$$

into the following immediate constituents:

left hand side p-tuple:

$$<X \rightarrow, X \rightarrow, A \rightarrow, A \rightarrow, B \rightarrow, B \rightarrow>$$

right hand side p-tuple:

$$<A, B, bb, bbA, bbb, bbbB>$$

The left hand side p-tuples of two grammars in the same shape class are not necessarily the same: there may be many different left hand side p-tuples associated with grammars in the same shape class. For instance, the grammar

$GRAMMAR_2$:

$$X \rightarrow A \mid B$$
$$B \rightarrow bb \mid bbA$$
$$A \rightarrow bbb \mid bbbB$$

is in the same shape class as $GRAMMAR_1$, but its left hand side p-tuple,

$$<X \rightarrow, X \rightarrow, B \rightarrow, B \rightarrow, A \rightarrow, A \rightarrow>$$

is different.

Let $Q = <q_1,\cdots,q_n>$ be a shape class. Define LPS to be the set of all the left hand side p-tuples of the grammars in Q. Similarly, define RPS to be the set of all the right hand side p-tuples of the grammars in Q. In GEN_1, Q is product-decomposed into LPS and RPS. As in any product-decomposition, GEN_1 generates the contents of LPS and RPS independently, and generates the grammars in Q by enumerating the product $LPS \times RPS$.

As produced by GEN_1, all the grammars in a given shape class, have the same left hand side p-tuple. That is, in GEN_1, LPS contains just one left hand side p-tuple. In the left hand side p-tuple GEN_1 uses for shape class $<q_1,\cdots,q_n>$, the first q_1 entries are the nonterminal N_1, the next q_2 entries are N_2, and so on. RPS contains all p-tuples consisting of right hand sides which are permitted in the current size class. Two p-tuples consisting of the same right hand sides in a different order are considered distinct.

4.2 An Alternative Generator of Context Free Grammars

The second generator described in [3], GEN_2 (originally called the "improved enumerator"), is similar to the first, in that both decompose the set of context free grammars into size classes, and the size classes into shape classes. In GEN_1, each grammar in a shape class is product-decomposed into two immediate constituents: a left hand side p-tuple and a right hand side p-tuple. In GEN_2, each grammar in a shape class is product-decomposed into n immediate constituents. Each immediate constituent consists of a distinct nonterminal N_i and a set containing the q_i right hand sides in the grammar which have N_i as their left hand side. For

instance, $GRAMMAR_1$ is product-decomposed by GEN_2 into three immediate constituents:

$$<X \rightarrow, \{A, B\}>$$
$$<A \rightarrow, \{bb, bbA\}>$$
$$<B \rightarrow, \{bbb, bbbB\}>$$

Said another way, the generative structure for shape class $Q = <q_1, \cdots, q_n>$ is

$$Q = (N_1, V_1) \times (N_2, V_2) \times \cdots \times (N_n, V_n)$$

where V_i is the collection of sets which contain q_i right hand sides. Each V_i is generated in a manner not unlike the manner in which RPS was generated by GEN_1.

4.3 Comparison of the Alterntive Generative Structures

One may wish to exclude from the complete set of context free grammars many different kinds of grammars. Characterizations of three different kinds of undesirable grammars are considered in this section. For each characterization, the minimum information required to apply the characterization is analyzed, and the availability of this information in the two generators is compared. In general, the generative structure associated with GEN_2 allows a much more effective use of *a priori* exclusion information than the generative structure associated with GEN_1.

4.3.1 Excluding Grammars Containing Multiple Copies of a Production Rule

Example:

$$X \rightarrow aaaX$$
$$X \rightarrow aaa$$
$$X \rightarrow aaa$$

contains two copies of a production rule.

The number of copies of a production rule in a grammar is defined in terms of the set of right hand sides associated with each left hand side in that grammar. Specifically, a grammar contains multiple copies of a production rule if any one of these sets contain two or more copies of the same right hand side. It is not necessary to know with which particular left hand side these sets are associated.

In the generator GEN_2 a set of right hand sides is associated with each left hand side of a production rule. At the time when a set of right hand sides is generated, it is easy for GEN_2 to assure that the set contains no duplicate entries. This effectively guarantees that no grammar contains multiple copies of any production rule.

By comparison, in GEN_1 the information necessary to recognize multiple copies of a production rule does not become available until the grammar is fully formed. It can therefore do no better than to use this characterization in the least effective manner, as an explicit test applied to each individual grammar after it has been completely generated.

4.3.2 Excluding Grammars Identical to Previously Generated Grammars Except for the Order of Production Rules

Example:

$$X \rightarrow aY$$
$$Y \rightarrow aZ$$
$$Z \rightarrow aX \mid a$$

is identical to

$$X \rightarrow aY$$
$$Y \rightarrow aZ$$
$$Z \rightarrow a \mid aX$$

except for the order of production rules.

One way to recognize that a particular set of productions has been generated previously in a different order is to define a canonical order on production rules and reject any grammar whose production rules are not in canonical order (or, generate only grammars whose rules are in canonical order).

One kind of canonical order for production rules is based on a fixed ordering of the nonterminals and a fixed ordering of all possible right hand sides. In this kind of canonical order, the productions whose left hand side is the first nonterminal occur first in the grammar, followed immediately by the productions whose left hand side is the second nonterminal, and so on. The order of productions with the same left hand sides is determined by the ordering of the possible right hand sides.

If the canonical order is of this kind, a generative structure in which the sets of right hand sides associated with each left hand side are constituents will enable grammars whose productions are not in canonical order to be effectively excluded.

4.3.3 Excluding Disconnected Grammars

A grammar is "connected" if all of its nonterminals are "reachable". A nonterminal is "reachable" if it is the initial nonterminal, or it is in the right hand side of a production whose left hand side is a nonterminal which is reachable. A grammar is "disconnected" if it is not connected, *i.e.* if one or more of its nonterminals is not reachable.

The following grammar is disconnected because Y is unreachable.

$$X \rightarrow aZ$$
$$Y \rightarrow aZ$$
$$Z \rightarrow aX \mid a$$

To determine if a grammar is connected, it is necessary to know for every nonterminal which nonterminals appear in the right hand sides of productions for which

that nonterminal is the left hand side. In both GEN_2 and GEN_1, this information does not become available until the product composition which finally produces whole grammars is in progress. In other words, the generative structures associated with both generators precludes large-scale exclusion of disconnected grammars.

5. General Discussion

Incremental learning systems are searching for a small target subspace of the large space of candidates. The information supplied by the performance assessor must be used to narrow and focus this search. This information is stated in terms which more or less directly refer to particular subspaces of candidate space. The thesis explored in this paper is that the effectiveness with which a learning system is able to make use of information is largely determined by how succinctly the subspace characterized by the information can be expressed in terms of the structure imparted on candidate space by the internal organization of the learning system.

This paper deals with a special case of the general thesis, looking only at the use of *a priori* exclusion information by enumerative systems. Enumerative systems are particularly illuminating because of the straightforward relation between the internal organization of enumerative systems and the structure this organization imparts on candidate space.

The assumptions about the information which are critical to the discussion in Section 2 are that it is specific (*i.e.* identifies a particular class of candidates) and definite (*i.e.* the candidates named are definitely not in the target set). The assumption that the information is *a priori* information is less of a restriction than it might at first seem. In hierarchical enumerative systems, the information which flows from a higher level generator to the generators immediately below is precisely *a priori* exclusion information: "Generate all and only objects of this kind". For instance, in the grammar generators in Section 3 the information characterizing a size class is *a priori* exclusion information for the shape class generator.

A few other researchers have looked at special cases of this general thesis. Holland [4] and Bethke [5] analyze the use of dynamic, probabilistic merit information by a specific family of systems, "genetic algorithms", in terms of a hierarchical union-decomposition of candidate space. Lenat and Brown [6] discusses the use of a similar kind of information by two specific systems, EURISKO and AM, and concludes that their success depends crucially on whether or not the structure imparted on candidate space by their internal representation language (and the syntactic operations the systems use) is a "useful" one.

These studies, like the study reported in this paper, conclude that the general thesis holds under certain conditions. Moreover, a concept similar to generative structure has played a central role in each study. Generative structure, and closely related concepts, are proving to be useful conceptual tools for analyzing the strengths and weakness of different kinds of learning systems.

References

[1] R.G. Smith, T.M. Mitchell, R.A. Chestek, and B.G. Buchanan, "A model for learning systems", *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (1977), pp. 338-343.

[2] R.K. Lindsay, B.G. Buchanan, E.A. Feigenbaum, and J. Lederberg, *Applications of Artificial Intelligence for Organic Chemistry: The Dendral Project,* McGraw-Hill (1980).

[3] R.M. Wharton, "Grammar enumeration and inference", *Information and Control,* (March, 1977), pp. 253-272.

[4] J.H. Holland, *Adaptation in Natural and Artificial Systems,* University of Michigan Press, Ann Arbour, (1975).

[5] A.D. Bethke, *Genetic Algorithms as Function Optimizers,* Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, (1981) (Ph.D. thesis).

[6] D.B. Lenat and J.S. Brown, "Why AM and Eurisko appear to work", *Artificial Intelligence,* (August, 1984), pp. 269-294.

DETECTING ANALOGOUS LEARNING

Ken Wellsch and Marlene Jones

Logic Programming and Artificial Intelligence Group
University of Waterloo
Waterloo, Ontario
CANADA

ABSTRACT

Analogy is both a common reasoning and instructional technique. There are several reasons why within an instructional or diagnostic system one would want to detect analogies; these are discussed herein. We present a particularly simple (the computational complexity is polynomial) but powerful algorithm for detecting analogies. We examine the effectiveness of this algorithm, which is based on subtree matching, as a means of detecting analogies within the context of instructional systems (such as ICAI systems). As an initial study in this regard, a particular framework for student modelling -- the genetic graph -- and a variety of examples from the domain of elementary ballet are employed as test cases. The results are discussed herein along with suggestions for future research.

1. Introduction

Reasoning by analogy is a common method of learning. It appears to be a technique that humans find easy to employ, even when little data is available; in fact, it is a technique which is often employed when nothing else works.

Many researchers both within the realm of Psychology and AI have investigated reasoning by analogy and techniques for detecting analogies. For a comprehensive survey of previous research, the reader should consult (Wellsch 1985). Within this paper, we restrict our attention to the use of analogy for instructional purposes, in particular, to the detection of analogous learning.

Analogy is a common instructional technique, as it is a means of getting the student to grasp concepts in a new domain without providing complex, technical descriptions within the new or *target* domain. For example, recall the story of Sir Isaac Newton and the falling apple. It was the fall of an apple that he claimed lead him to the discovery of the law of universal gravitation. The analogy he uses compares the earth and moon with the earth and an apple. Another common analogy, used within the context of chemistry, is that "the hydrogen atom is like the solar system". Based on this analogy and the facts that a planet resolves around the sun, that the sun is more massive than a planet, and that the nucleus is more massive than an electron, one is to conclude that an electron resolves around the nucleus. (This is one of the analogies used by Gentner in her structure-mapping approach to modelling analogies (Gentner 1982, 1983).) Analogies are employed by instructors within every domain; a common one when teaching the use of computer files is to refer to a computer file as one file within a filing cabinet (Halasz and Moran 1982) (This particular analogy has also been used as an illustration how the overgeneralization of an analogy can be harmful (Halasz and Moran 1982).)

Employing analogies also has the advantage of exploiting the student's previous knowledge, hence building on prior experience -- a sound pedagogical practice. Moreover, it is a technique that tends to produce fast results. An appropriate analogy provides sufficient insight into the new domain that one gets the desired 'eureka' response. The use of analogy has also been shown to be a valuable tool in helping student's develop problem solving skills, in particular, in detecting and generalizing the appropriate method of solving particular problems (Reed, Ernst and Banerji 1974). Studies have also shown that the use of analogies can be extremely helpful in the recall of learned material (Gick and Holyoak 1980, 1983; Ross 1984; Schustack and Anderson 1979). For example, individuals often recall a particular analogy and then employ this information to recall the desired fact or concept within the target domain.

Another advantage within the context of instruction is actually its use within diagnosis. Analogies are rarely perfect. Usually it is appropriate to map over only part of the data from the *base* domain to the target domain. A student may, however, attempt to employ all of the information from the base domain and hence extend an analogy inappropriately or perhaps is even employing an analogy which is totally inappropriate. Being able to detect which analogy is being used is an aid in diagnosing the student's problem and why the problem even arose. A simple illustration of this is when a student is learning a second language. To understand why a particular error pattern in sentence structure, location of phrases or verbs, use of prepositions etc., a good heuristic is to examine the analogous construct within the student's native language. By detecting the analogy that the student is employing one gains insight into the student's errors. This insight can then be incorporated into one's instruction with this student.

In addition to being both a powerful instructional and diagnostic technique, analogy is also a means of organizing instructional material. For example, if the instructor recognizes that certain concepts are analogous, these analogies can be exploited when teaching, not only by building upon the student's previous knowledge but by organizing the course material so that these analogies are apparent. However, it is also useful to know when (in a more general context) learning tasks or situations are analogous. For example, is the student being asked to apply the same technique but within a different domain? Or is some other aspect of the task similar? Can the learning tasks or situations be considered to be analogous? If so, then the student's previous successes or failures within the analogous situation can be and should be taken into account.

Goldstein (1982) felt that the use of analogy was of sufficient importance (even within the context of the simple game of WUMPUS) that he incorporated it as one of the types of information to be stored within his framework for student models -- the *genetic graph*. In fact, it is this framework that we employ here for the purpose of testing our algorithm within the context of instructional systems.

2. Student Models

Although several different approaches to modelling students have been proposed during the past decade, none is ideal. Usually, within a diagnostic or tutoring situation, there is a variety of information which one wants to represent with regard to the student's mastery of the domain:

(1) Knowledge (facts, concepts, procedural skills or whatever) that the student has mastered.

(2) Knowledge we know he/she has not mastered.

(3) Knowledge we believe to be mastered but for which we unfortunately do not have sufficient convincing evidence for, and most importantly.

(4) Any misconceptions, or procedural deviations.

In addition to domain-related information, one might also want to include in the student model data regarding the student's developmental history, learning preference etc. The type of information which one choses to represent depends both upon the individual student (for example, one needs to represent more information for a learning disabled student than an "average" student) and the learning situation (both the task and the domain should be taken into account).

There are several student modelling techniques which have been included in ICAI or diagnostic systems; for example, see (Burton 1982; Brown and Burton 1978; Carr and Goldstein 1977; Goldstein 1982; Jones and Poole, 1985; Palies et al. 1985; Reiser, Anderson and Farrell 1985; Schuster 1985; Westcourt, Beard and Gould 1977). For the purposes of the discussion here, we have selected one representation scheme for a student model, the *genetic graph* which was originally proposed by Goldstein (1982) and extended by Wasson (1985). There were many reasons for selecting this particular representation scheme including its flexibility, the fact that it is has been successfully employed in a variety of domains, and the fact that Goldstein's original design included the use of *analogy* links. Although we have initially restricted ourselves to testing our algorithm within one framework, the algorithm is not constrained to this one framework and could, in fact, be adapted for use within other student model representation schemes.

The *genetic graph* is a directed graph in which the nodes represent knowledge (facts, subskills and deviations thereof), the links represent relationships between the nodes (such as *generalizations, specializations, refinements, components, analogies, deviations, corrections, tests*). Moreover, the genetic graph can be viewed as a multi-dimensional structure which is divided into various *levels* of difficulty. This is a simple means of indicating the relative difficulty of subskills or concepts. Another useful means of organizing the graph's nodes is that of islands. An *island* is a cluster of nodes and the connecting links. Such a cluster may represent a single skill or body of knowledge which constitutes a concept. In order to represent both prerequisite skills and an ordering of steps within a procedural skill, *pre* and *post* links can be employed.

In other words, the genetic graph is a structure in which all desired information about the domain, deviations thereof, is stored. The student model is, in fact, an overlay on the genetic graph. One can view the genetic graph as a search space and the student model as a selection of appropriate pieces of the search space. The genetic graph approach has been successfully employed to model several diverse domains: the simple adventure game WUMPUS (Goldstein 1982), subtraction (Wasson 1985), elementary ballet (Wasson 1985), division (Dundas and Stockdale 1985). For further explanation of the genetic graph approach to student modelling, detailed examples, and a discussion of generating and maintaining the genetic graph, the reader should consult (Wasson 1985).

3. Representing the Genetic Graph

We wished to base the testing of our algorithm within the context of student models on previously published genetic graphs, rather than develop further examples which we might subconsciously tailor for success. Wasson's genetic graphs for the domain of elementary ballet were selected, because more information was available for these examples than those of the other previously published domains. (It is also a more interesting and challenging domain than WUMPUS, subtraction or division.) Sample genetic graphs for introductory ballet are illustrated in Figures 1 and 2. Each 'link' is labelled with a symbol which denotes its type:

C	*components,*
G	*generalization*
S	*specialization*
A	*analogy*
D	*deviation*
Corr	*correction.*

Figure 1 represents six basic stances or positions in ballet, partitioned into separate arm and leg positions. The *bras bas* is an arm position only, while first to fifth positions are positions for the entire body (see Figure 5).

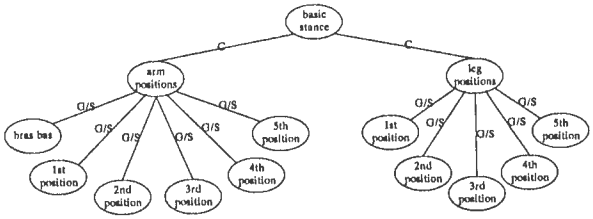


Figure 1
A subset of an introductory ballet genetic graph (Wasson 1985)

Figure 2 highlights a decomposition based on maturity and the sex of the ballet student, the hand and arm positions for a mature male. Notice that only first, second and fifth positions are presented; this is based on the teaching approach of (Lawson 1973). The various symbols used in Figure 2 have the following definitions:

curve 1	– follow the line of the shoulder, slightly downward
fingers 1	– fingertips level with the breastbone
fingers 2	– breadth of the forehead apart
hands not 2nd position	– downward according to line required
curve 2	– above the ears and by lifting the eyes eyes he can see the insides of his hands
fingers 3	– fingertips over and just in front of crown of head
fingers 4	– width of his forehead apart
hand 2nd position	– facing directly downward
shoulders	– pulled outward and pressed downward
chest	– fully expanded with easy breathing
handshake	– natural position
too stiff	– correct by softening

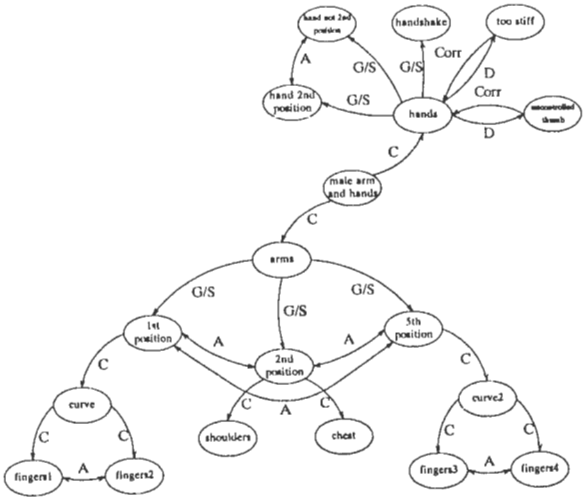


Figure 2
Another subset of an introductory
ballet genetic graph (Wasson 1985)

There are two distinct levels at which one can detect analogies within a genetic graph. The simplest level involves the individual nodes of the genetic graph. Detection of a strong analogy between two nodes in the genetic graph indicates that an *analogy link* should exist between these two nodes. The second level of representation is significantly more difficult to process as it involves the structure of the genetic graph itself. The objective is to find analogous subgraphs contained within the genetic graph.

It is first necessary to determine a computational representation for the genetic graph. We employ a common graph representation scheme, an *adjacency-vector / linked-list structure*. A *vertex*, which represents a node in the genetic graph, has a 'label', a pointer to the knowledge that it represents, and a pointer to a list of *edges*. An *edge* represents a link in the genetic graph. Each *edge* has a 'type' (*analogy, refinement* etc.), a pointer to an adjacent *vertex* (that the link is directed to), and a pointer to another *edge* structure (to form a linked-list of *edges*). The *adjacency vector* is a list that contains pointers to all of the vertices. There is only one physical structure created for each *vertex* , both the *adjacency vector* and any incoming *edge* have pointers to it. By indexing the adjacency vector, vertices can be directly accessed. To access the adjacent *vertices* of a *vertex* requires following the edge-linked-list for that *vertex* .

Secondly, we must determine an appropriate representation scheme for the knowledge represented at each node of the genetic graph. As the genetic graph structure does not put any restrictions on the representation of the knowledge contained at individual nodes, one may select the form of representation (frame, relational logic, network, etc.) which is best suited to the application domain at hand. For our purposes here, we represent the information at each node of the genetic graph as a forest (a collection of trees). This allows distinct, but related facts and/or procedures, to be represented at each node if desired. To illustrate this, consider the node which is to represent the location and characteristics of the arms when held in first position. In this case, the following facts are represented by a single tree:

First position, Arms -

- Both arms are forward of the body wall
- Both arms are parallel to the floor
- Both elbows are rounded moderately
- Both wrists are bend in
- The hands are separated by six inches
- The palms are open toward the face

which has the syntactic representation:

```
node (1st_position_arms)
[ 1st Position - Arms ]
{
    arms
    (
        direction(forward), parallel(floor),
        wrists (bent(in)),
        elbows (rounded(moderately)),
        hands
        (
            separation(6inches),
            palms (parallel(face))
        )
    )
}
```

4. The Analogy Algorithm

The significant role analogy plays in learning and instruction makes it necessary to include analogy within a student model, as seen with the genetic graph. Constructing a genetic graph for anything but exceedingly simple-minded domains will be tedious and error-prone, if done by hand, due to size and complexity. One step in automating the construction of a genetic graph would be by automated detection and inclusion of the *analogy links*.

The problem of detecting and applying analogies is very hard and although a great deal of work has been directed at analogical reasoning, it remains an open problem (for a fuller discussion of analogy and analogical reasoning, see (Wellsch 1985)). In the context of this paper, the subproblem of detecting analogies is tackled using a simple, but powerful polynomial algorithm.

The algorithm presented here is based on tree matching, which is an extension of the tree isomorphism problem. There are several reasons for employing a tree-like representation scheme for both the base and target domains. First there is evidence from various psychological studies to support the employment of a hierarchical structure (Bourne, Dominowski and Loftus 1979; Dember and Warm 1979; Reed and Flagg 1977). Secondly, the computational complexity of the resulting algorithms is an important factor. The algorithms which we develop for analogical reasoning are polynomial, whereas the analogous algorithms for the more general graph-based representation scheme are exponential.

When a tree is used to represent *knowledge*, the nodes of the tree correspond to objects and the relationships between the objects (i.e. the tree has labelled nodes). The tree isomorphism problem requires that the structure and labels be identical between the two trees (with branch permutations) for them to be isomorphic; a strict form of equality. On the other hand, an exact match between the base and target of an analogy (assuming one would call such a comparison an analogy) is of little value. To deviate from strict equality (to flexible equality) relies on some additional knowledge about the labels. Clearly, if our model of equality relies on labels being identical for equality and otherwise being not equal, then our model of analogy could not function. The background information regarding labels provides a means of judging the *similarity* of two labels (not just "equal" or "not equal"); strict equality is inadequate in this situation.

The area of judging similarity has received a reasonable amount of attention (Ortony 1979; Tversky 1977). Although the method used here does not attempt to achieve the same level of sophistication as today's similarity models, it has proved adequate. The relationships between the various tree labels of the domain trees are represented in a single hierarchy called a *background knowledge tree (BKT)* (see Figure 3). The *similarity function* σ given two labels, computes a value from the relative positions of the two labels within the BKT, i.e. a measure of distance. Greater "dis-

tance" implies lower similarity between the two labels. This form of measure is certainly not ideal, but whatever metric is chosen, as long as it yields values such that a larger value signifies lower similarity, then the exact definition of σ will not effect the matching algorithm.

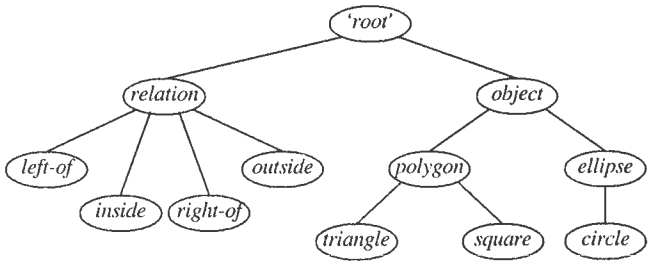


Figure 3
General background knowledge hierarchy (BKT)

Returning once again to the idea of tree matching, given the function σ, one can develop an algorithm of "flexible" matching for trees. There are a variety of ways of combining the values of individual node pairings (obtained from σ) to judge the similarity of two trees. For example, the worst node-pair rating, the best node-pair rating, and the sum of all the node-pair ratings (cost) are three such measures; others certainly exist. In the research discussed here, the worst node-pair rating and the total "cost" are used to judge tree similarity. Matching two trees t_1 and t_2 yields the ordered pair $<cost(t_1,t_2),worst(t_1,t_2)>$. Notice that the *cost* function pairs the most appropriate (least costly) branches, and the overall cost is penalized for any remaining unmatched branches (the *cost* function presented here assumes, without loss of generality, that $|t_1| \leq |t_2|$).

Consider now the subtree isomorphism problem: given two trees, t_1 and t_2, is one tree isomorphic to a subtree of the other? Note that a subtree s of a tree t can be the tree t itself (i.e. $s=t$) and that we are not considering the problem of matching **all** the subtrees of t_1 with all the subtrees of t_2 (That would be grim indeed!). Here we have a selection problem: which subtree of t_1 *best* matches with t_2 as given by the $<cost,worst>$ rating? The selection strategy used is one that picks the matching with the lowest worst node-pair rating (i.e minimize worst node-pair rating), and if there is more than one that has this minimum rating, then break such a tie by minimizing the cost rating among the minimum worst node-pair matchings (i.e pick the matching with the lowest cost rating). Subtree matching consists of selecting the "best" subtree pairing.

Finally, at the level of the base and target representation, is *forest* matching. Both the base and target domains are represented by forests, i.e. sets of trees. It is at this level of representation that we refer to matching as that associated with analogy. Forest matching uses the same basic approach taken with subtree matching (i.e. the selection technique) but with a few twists. Things become a bit more complicated because the trees in a domain forest are significantly more independent than the nodes in a tree (other than pertaining to the same domain, there isn't necessarily any more of a connection). This structure independence (and common-sense) implies that combining an unequal number of structures (trees) taken from one domain with another has no basis. The natural approach might be to simply eliminate each of the already paired trees (obtained from "best" matching) from consideration and use only the remainder for later pairings (i.e. a process of elimination).

Two obvious problems can occur using such a matching algorithm. The first is *multiplicity*, i.e. a one-to-many mapping of an object or relation to another. In terms of a tree representation, this manifests itself when a single leaf node is paired to a subtree composed of more than one node. The second problem is *inconsistency*, i.e. a mapping that contains an object or relation that seems to map to more than one distinct object or relation (or visa versa). This is much like multiplicity, but simpler to deal with than multiplicity. The algorithm penalizes a matching that results in inconsistent mappings (in function σ). The algorithm (in Figure 4) recognizes with both these problems (although there are probably other approaches to dealing with multiplicity).

$$cost(t_1,t_2) \equiv \text{if } root(t_1) = root(t_2) = \text{nil then } 0$$
$$\text{else if } root(t_1) = \text{nil or } root(t_2) = \text{nil or } \{leaf(t_1) \oplus leaf(t_2)\} \text{ then}$$
$$\text{LARGE-CONSTANT} \times (|t_1| + |t_2|)$$
$$\text{else } \sigma(root(t_1),root(t_2)) + \sum_{i=1}^{\delta(root(t_1))} \underset{j=l}{\overset{\delta(root(t_2))}{\text{MIN}}} \left\{ cost(child(i,t_1),child(j,t_2)) \right\}$$
$$+ \text{LARGE-CONSTANT} \times \sum_{j=\delta(root(t_1))+1}^{\delta(root(t_2))} |child(j,t_2)| \text{ endif.}$$

$$worst(t_1,t_2) \equiv \text{if } root(t_1) = root(t_2) = \text{nil then } 0$$
$$\text{else if } \delta(root(t_1)) \neq \delta(root(t_2)) \text{ then LARGE-CONSTANT}$$
$$\text{else } \underset{i=1}{\overset{\delta(root(t_1))}{\text{MAX}}} \left\{ \sigma(root(t_1),root(t_2)),\ worst(child(i,t_1),child(i,t_2)) \right\} \text{ endif.}$$

$$pair\text{-}min\ (<cost_1,worst_1>,<cost_2,worst_2>) \equiv \text{if } (worst_1 < worst_2) \text{ or}$$
$$((worst_1 = worst_2) \text{ and } (cost_1 < cost_2)) \text{ then } <cost_1,worst_1>$$
$$\text{else } <cost_2,worst_2> \text{ endif.}$$

$$\sigma(l_1,l_2) \equiv \text{if } inconsistent(l_1,l_2) \text{ then LARGE-CONSTANT}$$
$$\text{else } (|depth(l_1) - depth(l_2)| + \text{MAX}(|depth(l_1) - depth(\rho)|,$$
$$|depth(l_2) - depth(\rho)|)) \times \text{MAX}(priority(l_1),priority(l_2)) \text{ endif.}$$
$$where$$
$$depth(n) \text{ is the depth in the background knowledge hierarchy of node } n \text{ and } \rho \text{ is the common ancestor of } l_1 \text{ and } l_2.$$

$$tree\text{-}match\ (t_1,t_2) \equiv <cost(t_1,t_2),worst(t_1,t_2)>.$$

$$subtree\text{-}match\ (t_1,t_2) \equiv \underset{t \text{ subtree of } t_1}{pair\text{-}min} \left\{ tree\text{-}match\ (t,t_2) \right\}.$$

$$forest\text{-}match\ (f_1,f_2) \equiv \underset{t_i \in f_1}{pair\text{-}min} \left\{ \underset{\text{eliminate } t_j \text{ from } f_2}{subtree\text{-}match}\ (t,t_i) \mid t_j \in f_2 \right\}.$$

Figure 4
Matching algorithm

The algorithm described here was first thoroughly tested using problems from the domain of two-dimensional geometric shapes -- such geometric-analogies are often common problems on intelligence tests -- before being extended for use within the context of instructional systems. For a more thorough description of the algorithms, proofs of their complexity bounds, a discussion of the test examples and the results, see (Wellsch 1985).

5. Testing

Applying the algorithm to the problem of detecting analogies for the automated construction of genetic graphs is a major task. The testing that has been carried-out to date for this particular application has been limited to analogy detection within a subset of knowledge associated with introductory ballet instruction. We describe here one particular test case; for a discussion of other test cases see (Wellsch 1985).

There are six basic body stances or positions that are taught to introductory ballet students (see Figure 5). The *bras bas* is a basic arm position; the arm motion to first position consists of moving the arms upward from *bras bas*.

The algorithm was first applied to the overall physical characteristics shown in Figure 5, arm and leg positions are combined. Table 1A and 1B show the results. All of the possible pairings resulted in

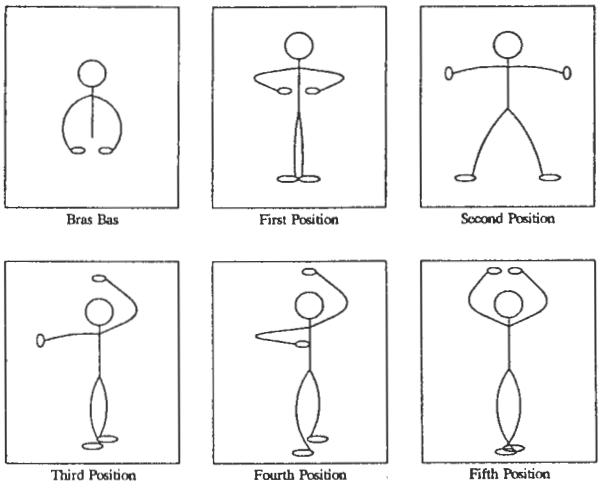


Figure 5
Six basic body stances for ballet

a worst-case of LARGE-CONSTANT so the *cost* values are the only way of differentiating between them. There are two points to be made about the numbers that appear in Table 1A. First, they represent the accumulated ratings for each node pairing (or individual node when no pairing was possible). Thus they are a function of the number of nodes used in the trees.

The second point is the basis by which the numbers are computed. The constant LARGE-CONSTANT was assigned a value of 100. The remaining values that compose the *cost* are the rated *similarity* for each node pairing. Conveniently enough, one can distinguish between the node similarity ratings and the number of occurrences of LARGE-CONSTANT in the raw cost values in Table 1A. For example, the matching between *first position* and *second position* has a raw cost of 703. This suggests that there were 7 nodes that could not be matched, and those that did match had a total difference of 3.

Table 1A
Overall cost values (raw)

cost values	Bras Bas	1st position	2nd position	3rd position	4th position	5th position
Bras Bas	0	200	903	1508	1508	400
1st pos.	200	0	703	2818	2517	1108
2nd pos.	903	703	0	2615	2825	1110
3rd pos.	1508	2818	2615	0	706	2150
4th pos.	1508	2517	2825	706	0	2031
5th pos.	400	1108	1110	2150	2031	0

Table 1B
Overall cost values (average)

cost values	Bras Bas	1st position	2nd position	3rd position	4th position	5th position
Bras Bas	0	4	18	24	25	9
1st pos.	4	0	11	36	33	18
2nd pos.	18	11	0	33	36	18
3rd pos.	24	36	33	0	7	29
4th pos.	25	33	36	7	0	27
5th pos.	9	18	18	29	27	0

Table 2

Interesting Pairings			
Position	Average Cost	Value	Position
Bras Bas	5	100	1st position arms
Bras Bas	12	100	5th position arms
1st position arms	18	100	2nd position arms
1st position arms	18	100	5th position arms
1st position legs	0	1	2nd position legs
2nd position arms	17	100	5th position arms
3rd position arms	11	100	4th position arms
3rd position arms	43	2	5th position arms
3rd position legs	0	1	4th position legs
3rd position legs	3	100	5th position legs
4th position arms	41	2	5th position arms
4th position legs	3	100	5th position legs

The values in Table 1B represent the average computed by dividing the raw score (from Table 1A) by the number of nodes involved. If the maximum value (that a node can be rated at) is one hundred, the average values in Table 1B can range from zero to one hundred inclusive. Because this average is independent of the number of nodes involved, it can then be used to compare the different pairings.

The results in Table 1(A&B) are in keeping with intuition. *Bras bas* is quite similar to *first position*. The fact that *bras bas* is a position variation on the arm position of *first position* suggests that this a reasonable result. *First position* is similar to *second position* and to a lesser extent, *fifth position*; the arms in these three are symmetric. The leg position of *fifth position* accounts for the decrease in similarity with both first and second. *Third position* appears to correspond best with *fourth position*; they both share an asymmetry in the arm positions. Hence the results of the algorithm agree with an intuitive view of the six stances.

A further refinement was tested that separates the arm and leg positions. Tables 3A, 3B and 4 contain the results of all possible pairings. Again, the results are intuitively acceptable. Table 3A and 3B contain the total *cost* values for the detailed matching. The values have the same basis as those in Table 1A and 1B. The values in Table 4 are the *worst-case values* for the matching. The '*' entries represent a worst-case occurrence of LARGE-CONSTANT. Since one cannot assign a value worse than this, the *cost* value is the only way to distinguish between these cases. It is the matches that have a worst-case less than LARGE-CONSTANT for which this value has differentiating capability. (Note that an exact match has a worst-case value and cost of zero.) When the *worst-case value* is less than LARGE-CONSTANT, there is an isomorphic matching of the two tree structures and the node symbols differ by at worst, that value.

Looking at Tables 3A, 3B and 4, there are twelve interesting matchings. They stand out from the other matchings, either because they have a low cost relative to the other matchings, or that they have a worst-case value that is quite small (i.e. < LARGE-CONSTANT). These matchings are summarized in Table 2. In this example, costs below 20 were considered interesting. Whether an absolute value (such as 20) exists for most or all cases or must be determined for each domain, is an open problem.

Again *bras bas* matches well with both first and fifth positions. *First position - arms* matches well with second and fifth position arms, but it is the *first position - legs* match with *second position - legs* that is interesting. *Third position - legs* and *fourth position - legs* matching also shares this strong similarity. The effect is due to the fact that each pair differs only in the separation of the feet. Otherwise, they are the same.

Table 3A
Detailed Cost values (raw)

cost values		Bras Bas	1st arms	1st legs	2nd arms	2nd legs	3rd arms	3rd legs	4th arms	4th legs	5th arms	5th legs
Bras Bas		0	200	1504	903	1504	2413	1508	2209	1508	400	1508
1st	arms	200	0	1704	702	1704	2310	1708	2009	1708	600	1708
	legs	1504	1704	0	1509	1	2877	508	2677	508	1407	508
2nd	arms	903	702	1509	0	1509	2107	1609	2317	1609	602	1609
	legs	1504	1704	1	1509	0	2877	508	2677	508	1407	508
3rd	arms	2413	2310	2877	2107	2877	0	2677	705	2677	2049	2772
	legs	1508	1708	508	1609	508	2677	0	2577	1	1606	101
4th	arms	2209	2009	2677	2317	2677	705	2577	0	2577	1929	2672
	legs	1508	1708	508	1609	508	2677	1	2577	0	1606	102
5th	arms	400	600	1407	602	1407	2049	1606	1929	1606	0	1606
	legs	1508	1708	508	1609	508	2772	101	2672	102	1606	0

Table 3B
Detailed Cost values (average)

cost values		Bras Bas	1st arms	1st legs	2nd arms	2nd legs	3rd arms	3rd legs	4th arms	4th legs	5th arms	5th legs
Bras Bas		0	5	51	25	51	50	50	47	50	12	50
1st	arms	5	0	54	18	54	46	53	41	53	18	53
	legs	51	54	0	45	0	63	18	60	18	50	18
2nd	arms	25	18	45	0	45	40	47	45	47	17	47
	legs	51	54	0	45	0	63	18	60	18	50	18
3rd	arms	50	46	63	40	63	0	58	11	58	43	60
	legs	50	53	18	47	18	58	0	57	0	55	3
4th	arms	47	41	60	45	60	11	57	0	57	41	59
	legs	50	53	18	47	18	58	0	57	0	55	3
5th	arms	12	18	50	17	50	43	55	41	55	0	55
	legs	50	53	18	47	18	60	3	59	3	55	0

Table 4
Worst-case values

worst-case values		Bras Bas	1st arms	1st legs	2nd arms	2nd legs	3rd arms	3rd legs	4th arms	4th legs	5th arms	5th legs
Bras Bas		0	•	•	•	•	•	•	•	•	•	•
1st	arms	•	0	•	•	•	•	•	•	•	•	•
	legs	•	•	0	•	1	•	•	•	•	•	•
2nd	arms	•	•	•	0	•	•	•	•	•	•	•
	legs	•	•	1	•	0	•	•	•	•	•	•
3rd	arms	•	•	•	•	•	0	•	•	•	2	•
	legs	•	•	•	•	•	•	0	•	1	•	•
4th	arms	•	•	•	•	•	•	•	0	•	2	•
	legs	•	•	•	•	•	•	1	•	0	•	•
5th	arms	•	•	•	•	•	2	•	2	•	0	•
	legs	•	•	•	•	•	•	•	•	•	•	0

• indicates that multiplicity or inconsistency occurred

Both *third position - legs* and *fourth position - legs* share a similarity with *fifth position - legs*. This similarity is weaker than the two previously described leg matchings because they share the more general notion of one foot in front of the other. An interesting matching exists between *third position - arms* and *fifth position - arms*, and *fourth position - arms* and *fifth position - arms*. The matching costs are both very high, yet their worst-case values are very low. This would seem to be an odd situation but, in fact, does have a simple explanation. Consider Figure 5; both third and fourth position have one arm the same as in fifth position. This leads to a strong matching. It is the lack of a match for the other arm in both third and fourth positions that results in such a high cost.

6. Concluding Remarks

Although the initial testing has been limited, the results have been encouraging. As illustrated by the test cases presented in the previous section, the results do correspond with one's intuition i.e. the analogies detected by the algorithm are easily explained. This is also true of the numerous test cases we employed within the domain of two-dimensional geometric shapes; the results of which are discussed in (Wellsch 1985). At this point the full power of the algorithm has not been investigated; the analogies we have investigated in the context of instructional systems are as yet restrictive in nature. As mentioned in the opening remarks, one would like to be able to detect analogous learning situations (including concepts, techniques, instructional methodologies, examples), so that this knowledge can then be employed to improve the diagnostic and instructional capabilities of a system. We have not yet applied our algorithm to such extensive examples, but then again such extensive student models have not yet been developed for *any* domain.

One of the potential advantages of an effective algorithm for detecting analogies is in facilitating the development of dynamic student models. Generating a student model (regardless of the chosen representation scheme) can be a slow and tedious task, particularly if it includes the desired features mentioned in Section 2. Automating the creation of a student model (or parts thereof) given some domain knowledge is a worthwhile goal. Moreover, one does not want a static student model; after all large chunks of it may be unnecessary for a particular student. Rather, student models should be dynamic. (This is especially true when working with a variety of student populations or special populations such as learning disabled students where one expects more inter-student variation). The development of dynamic student models is a major open research issue within the area of ICAI. Creating and maintaining the genetic graph by an automated process is one important aspect of a dynamic student model which is then formed by an overlay on the genetic graph. One of the tasks necessary for such automation is the determination of the *analogy links* within the genetic graph.

During the course of this investigation, many interesting issues, both in regard to analogy detection and student modelling, have risen. For example, how much information should be represented at a genetic graph node? ICAI research has not yet addressed this question, either in the context of the genetic graph or any other student model. Because the algorithm is not solely based on a node-to-node matching, altering the amount of knowledge stored at a node should not effect the performance of the algorithm, but we have not verified this.

With regard to more general issues concerning analogy testing, the handling of *inconsistency* and *multiplicity* warrant further investigation. Again our initial results within both the domains of two-dimensional geometric shapes and genetic graphs, are very encouraging; the current approach (although simple) appears to be very effective. Although a more sophisticated method of handling these issues *may* improve applicability, one does not want to increase the computational complexity of the algorithms (which is polynomial in the size of the trees).

7. References

Adelman, H.S. (1982), "Identifying learning problems at an early age: A critical appraisal", *Journal of Clinical Child Psychology, 11*, 255-261.

Bourne, L. E., Domonowski, R. L. and Loftus, E. F. (1979), *Cognitive Processes,* Prentice-Hall.

Brown, J.S. and Burton, R.R. (1978), "Diagnostic Models for Procedural Bugs in Basic Mathematical Skills", *Cognitive Science, 2*, 155-192.

Burton, R.R. (1982), "Diagnosing Bugs in a Simple Procedural Skill", in D. Sleeman and J.S. Brown (eds), *Intelligent Tutoring Systems*, Academic Press, 157-183.

Carr, B.P. and Goldstein, I.P. (1977), "Overlays: A Theory of Modelling for Computer Aided Instruction", MIT Technical Report 210.

Dember, W. N. and Warm, J. S. (1979), *Psychology of Perception,* Holt, Rinehart and Winston.

Dundas, M. and Stockdale, G. (1985), "A Student Model for Long Division using the Genetic Graph Approach", CS486 course project, unpublished manuscript.

Gentner, D. (1982), "A Structure-Mapping Approach to Analogy and Metaphor", *IEEE Proceedings, International Conference on Cybernetics and Society,* 75-79.

Gentner, D. (1983), "Structure Mapping: A Theoretical Framework for Analogy", *Cognitive Science*, 7, 155-170.

Gick, M. L. and Holyoak, K. J. (1980), "Analogical Problem Solving", *Cognitive Psychology*, 12, 306-355.

Gick, M. L. and Holyoak, K. J. (1980), "Schema Induction and Analogical Transfer", *Cognitive Psychology*, 15, 1-38.

Goldstein, I.P. (1982), "The Genetic Graph: A Representation for the Evolution of Procedural Knowledge", in D. Sleeman and J.S. Brown (eds), *Intelligent Tutoring Systems*, Academic Press, 51-77.

Halasz, F. and Moran, T. P. (1982), "Analogy Considered Harmful", *Proceedings, Human Factors in Computer Systems*, 383-386.

Jones, M. and Poole, D. (1985), "An Expert System for Educational Diagnosis Based on Default Logic", *Proceedings of the Fifth International Conference on Expert Systems and Their Applications*, Avignon, France, 673-683.

Lawson, J. (1973), *The Teaching of Classical Ballet: Common Faults in Young Dancers and Their Training*, A&C Black Limited.

Ortony, A. (1979), "Beyond Literal Similarity", *Psychological Review*, 86, 161-180.

Palies, O., Caillot, M., Cauzinille-Marmeche, E., Lauriere, J-L., and Mathieu, J. (1985), "Student Modelling by an Expert System in an Intelligent Tutoring System", preprint.

Reed, S.K., Ernst, G.W. and Banerji, R. (1974), "The Role of Analogy in Transfer Between Similar Problem States", *Cognitive Psychology 6*, 436-450.

Reiser, B.J., Anderson, J.R. and Farrell, R.G. (1985), "Dynamic Student Modelling in an Intelligent Tutor for Lisp Programming", *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, Los Angeles, 8-14.

Reynolds, A. G. and Flagg, P. W. (1977), *Cognitive Psychology*, Winthrop.

Ross, B. H. (1984), "Reminding and Their Affects in Learning a Cognitive Skill", *Cognitive Psychology*, 16, 371-416.

Schustack, M. W. and Anderson, J. R. (1979), "Effects of Analogy on Prior Knowledge on Memory for New Information", *Journal of Verbal Learning and Verbal Behavior*, 18, 565-583.

Schuster, E. (1985), "Grammars as User Models", *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, Los Angeles, 20-22.

Tversky, A. (1977), "Features of Similarity", *Psychological Review*, 84, 327-352.

Wasson, B.J. (1985), *The Development of Student Models: The Genetic Graph Approach*, M.Math Thesis, Department of Computer Science, University of Waterloo; also available as Research Report CS-85-10, 102p.

Wellsch, K. C. (1985), *A Computational Model for Reasoning by Analogy*, M. Math Thesis, Department of Computer Science, University of Waterloo; also available as Research Report CS-85-19, 167p.

Westcourt, K., Beard, M. and Gould, L. (1977), "Knowledge-based Adaptive Curriculum Sequencing for CAI: Application of a Network Representation", *Proceedings of 1977 Annual Conference, Association for Computing Machinery*, 234-240.

$GUMS_1$: A General User Modeling System

Tim Finin
Computer and Information Science
University of Pennsylvania
Philadelphia, PA

David Drager
Arity Corporation
Concord, MA

Abstract

This paper describes a general architecture of a domain independent system for building and maintaining *long term models* of *individual users*. The user modeling system is intended to provide a well defined set of services for an *application system* which is interacting with various users and has a need to build and maintain models of them. As the application system interacts with a user, it can acquire knowledge of him and pass that knowledge on to the user model maintenance system for incorporation. We describe a prototype *general user modeling system* (hereafter called $GUMS_1$) which we have implemented in Prolog. This system satisfies some of the desirable characteristics we discuss.

Introduction - The Need for User Modeling

Systems which attempt to interact with people in an intelligent and cooperative manner need to know many things about the individuals with whom they are interacting. Such knowledge can be of several different varieties and can be represented and used in a number of different ways. Taken collectively, the information that a system has of its users is typically refered to as its *user model*. This is so even when it is distributed through out many components of the system.

Examples that we have been involved with include systems which attempt to provide help and advice [4, 5, 15], tutorial systems [14], and natural language interfaces [16]. Each of these systems has a need to represent information about individual users. Most of the information is acquired incrementaly through direct observation and/or interaction. These systems also needed to infer additional facts about their users based on the directly acquired information. For example, the WIZARD help system [4, 15] had to represent which VMS operating system objects (e.g. commands, command qualifiers, concepts, etc) a user was familiar with and to infer which other objects he was likely to be familiar with.

We are evolving the e design of a general user model maintenance system which would support the modeling needs of the projects mentioned above. The set of services which we envision the model maintenance system performing includes:

- maintaining a data base of observed facts about the user.
- infering additional true facts about the user based on the observed facts.
- infering additional facts which are likely to be true based on default facts and default rules.
- informing the application system when certain facts can be infered to be true or assumed true.
- maintaining the consistency of the model by retracting default information when it is not consistent with the observed facts.
- providing a mechanism for building hierarchies of *stereotypes* which can form initial, partial user models.
- recognizing when a set of observed facts about a user is no longer consistent with a given stereotype and suggesting alternative stereotypes which are consistent.

This paper describes a general architecture for a domain independent system for building and maintaining *long term models* of *individual users*. The user modeling system is intended to provide a well defined set of services for an *application system* which is interacting with various users and has a need to build and maintain models of them. As the application system interacts with a user, it can acquire knowledge of him and pass that knowledge on to the user model maintenance system for incorporation. We describe a prototype *general user modeling system* (hereafter called $GUMS_1$) which we have implemented in Prolog. This system satisfies some of the desirable characteristics we discuss.

What is a User Model?

The concept of encorporating user models into interactive systems has become common, but what has been meant by a user model has varied and is not always clear. In trying to specify what is being refered to as a user model, one has to answer a number of questions: who is being modeled; what aspects of the user are being modeled; how is the model to be initially acquired; how will it be maintained; and how will it be used. In this section we will attempt to characterize our own approach by answering these questions.

Who is being modeled?

The primary distinctions here are whether one is modeling individual users or a class of users and whether one is attempting to construct a short or long term model. We are interested in the aquisition and use of long term models of individual users. We want to represent the knowledge and beliefs of individuals and to do so in a way that results in a persistent record which can grow and change as neccessary.

It will be neccessary, of course,to represent generic facts which are true of large classes (even all) of users. In particular, such facts may include inference rules which relate a person's belief, knowledge or understanding of one thing to his belief, knowledge and understanding of others. For example in the context of a timeshared computer system we may want to include a rule like:

> *If a user U believes that machine M is running, then U will believe that it is possible for him to log onto M.*

It is just this sort of rule which is required in order to support the kinds of cooperative interactions studied in [6] and [7], such as the following:

```
User:   Is UPENN-LINC up?

System: Yes, but you can't log on now.
        Preventative maintenance is being
        done until 11:00am.
```

What is to be modeled?

Our current work is focused on building a general purpose, domain independent model maintenance system. Exactly what information is to be modeled is up to the application. For example, a natural language system may need to know what language terms a user is likely to be familiar with [16], a CAI system for second language learning may need to model a user's knowledge of grammatical rules [14], an intelligent database query system may want to model which fields of a data base relation a user is interested in [10], and an expert system may need to model a user's domain goals [11].

How is the model to be aquired and maintained?

We are exploring a system in which an initial model of the user will be selected from a set of stereotypical user models [13]. Selecting the most appropriate stereotype from the set can be accomplished by a number of techniques, from letting the user select one to surveying the user and having an expert system select one. Once an initial model has been selected, it will be updated and maintained as direct knowledge about the user is aquired from the interaction. Since the use of stereotypical user models is a kind of *default reasoning* [12], we will use *truth maintenance* techniques [9] for maintaining a consistent model.

In particular, if we learn something which contradicts a fact in the our current model of the user than we need to update the model. Updating the model may lead to an inconsistency which must be squared away. If the model can be made consistent by changing any of the *default* facts in the model, then this should be done. If there is a choice of which defaults to alter, then a mechanism must be provided to do this (e.g. through further dialogue with the user). If there are no defaults which can be altered to make the model consistent then the stereotype must be abandoned and a new one sought.

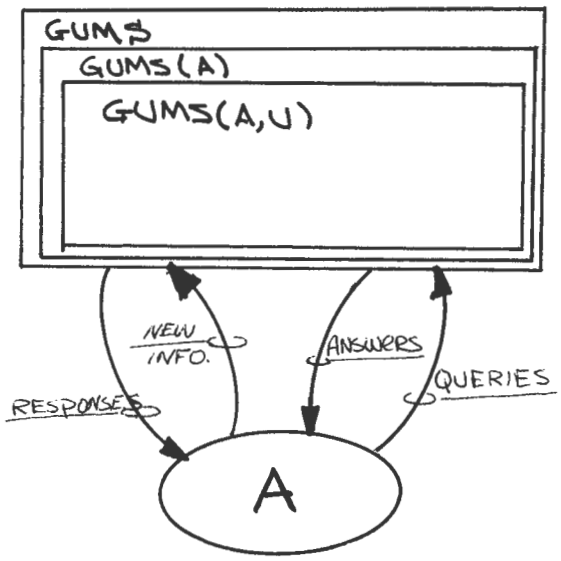
How is the model to be used?

The model can be accessed in two primary ways: facts can be added, deleted or updated from the model and facts can be looked up or infered. A forward chaining component together with a truth maintenance system can be used to update the default assumptions and keep the model consistent.

Architectures for User Modeling Systems

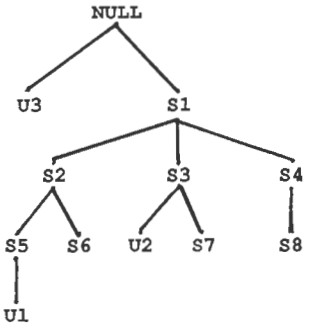
Our goal is to provide a general user modeling utility organized along the lines shown in figures 1 and 2. The user modeling system provides a service to an application program which interacts directly with a user. This application program gathers information about the user through this interaction and choses to store some of this information in the user model. Thus, one service the user model provides is accepting (and storing!) new information about the user. This information may trigger an inferential process which could have a number of outcomes:

- The user modeling system may detect an inconsistency and so inform the application.
- The user model may infer a new fact about the user which triggers a demon causing some action (e.g. informing the application).



A: an Application
GUMS: General User Modeling System
GUMS(A): Modeling System for Application A
GUMS(A,U): Model for User U in Application A

Figure 1: A General Architecture for a User Modeling Utility



NULL: the Empty Stereotype
Si: Stereotype i
Ui: User i

Figure 2: A User Modeling System for an Application

- The user model may need to update some previously infered default information about the user

Another kind of service the user model must provide is answering queries posed by the application. The application may need to look up or deduce certain information about its current user.

We are currently experimenting with some of these ideas in a system called $GUMS_1$. This system is implemented in prolog and used a simple default logic together with a backward chaining interpreter rather than a truth maintenance system and a forward chaining engine. The next section describes $GUMS_1$ and its use of default logic.

Default Logic and User Modeling

A user model is most useful in a situation where the application does not have complete information about the knowledge and beliefs of its users. This leaves us with the problem of how to model a user given we have only a limited amount of knowledge about him. Our approach involves using several forms of default reasoning techniques: stereotypes, explicit default rules, and failure as negation.

We assume that the $GUMS_1$ system will be used in an application which incrementaly gains new knowledge about its users throughout the interaction. But the mere ability to gain new knowledge about the user is not enough. We can not wait until we have full knowledge about a user to reason about him. Fortunately we can very often make generalization about users or classes of users. We call a such a generalization a stereotype. A stereotype consists of a set of facts and rules that are believed to applied to a class of users. Thus a stereotype gives us a form of default reasoning.

Stereotypes can be organized in hierarchies in which one stereotype subsumes another if it can be thought to be more general. A stereotype S_1 is said to be more general than a stereotype S_2 if everything which is true about S_1 is neccessarily true about S_2. Looking at this from another vantage point, a stereotype inherits all the facts and rules from every stereotype that it is subsumed by. For example, in the context of a *programmer's apprentice* application, we might have stereotypes corresponding to different classes of programmer, as is suggested by the the hierarchy in figure 2.

In general, we will want a stereotype to have any number of immediate ancestors, allowing us to compose a new stereotype out of several existing ones. In the context of a *programmers apprentice*, gor example, we may wish to describe a particular user as a *SymbolicsWizard* and a *UnixNovice* and a *ScribeUser*. Thus, the stereotype system should form a general lattice. Our current system constrains the system to a tree.

Within a stereotype we can have default information as well. For instance, we can be sure that a programmer will know what a *file* is, but we can only guess that a programmer will know what a *file directory* is. If we have categorized a given user under the *programmer stereotype* and discover[1] that he is not familiar with the concept of a *file* then we can concludethat we had improperly chosen a stereotype and must choose a new one. But if we got the information that he did not know what a *file directory* was, this would not rule out the possibility of him being a programmer. Thus $GUMS_1$

[1]perhaps through direct interaction with her

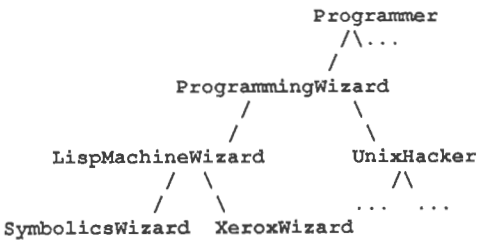


Figure 3: A Hierachy of Stereotypes

allows rules and facts within a stereotype to be either definitely true or true by default (i.e. in the absence of information to the contrary.)

In $GUMS_1$ we use the **certain/1** predicate to introduce a definite fact or rule and the **default/1** predicate to indicate a default fact or rule, as in:

certain(P).	a definite fact: P is true.
certain(P if Q).	a definite rule: P is true if Q is definitely true and P is assumed to be true if Q is only assumed to be true.
default(P).	a default fact: P is assumed to be true unless it is known to be false.
default(P if Q).	a default rule: P is assumed to be true if Q is true or assumed to be true and there is no definite evidence to the contrary.

As an example, consider a situation in which we need to model a persons familiarity with certain terms. This is a common situation in systems which need to produce text as explanations or in response to queries and in which there is a wide variation in the users' familiarity with the domain. We might use the following rules

(a) default(understandsTerm(ram)).

(b) default(understandsTerm(rom)
if understandsTerm(ram)).

(c) certain(understandsTerm(pc)
if understandsTerm(ibmpc)).

(d) certain(~understandsTerm(cpu)).

to represent these assertions, all of which are considered as pertaining to a particular user with respect to the stereotype containing the rules:

(a) Assume the user understands the term *ram* unless we know otherwise.

(b) Assume the user understands the term *rom* if we know or believe he understands the term *ram* unless we know otherwise.

(c) This user understands the term *pc* if he understands the term *ibmpc*.

(d) This user does understand the term *cpu*.

$GUMS_1$ also treats negation as failure in some cases as a default rule. In general, logic is interpreted using an open world assumption. That is, the failure to be able to prove a proposition is not taken as evidence that it is not true. Many logic programming languages, such a prolog, encourage the interpretation of unprovability as logical negation. Two approaches have been forwarded to justify the

negation as failure rule. One approach is the closed world assumption [2]. In this case we assume that anything not inferable from the database is by necessity false. One problem with this assumption is that this is a metalevel assumption and we do not know what the equivalent object level assumptions are. A second approach originated by Clark is based upon the concept of a completed database [1]. A completed database is the database constructed by rewriting the set of clauses defining each predicate to an **if and only if** definition that is called the completion of the predicate. The purpose of the completed definition is to indicate that the clauses that define a predicate define every possible instance of that predicate.

Any approach to negation as failure requires that a negated goal be ground before execution, (actually a slightly less restrictive rule could allow a partially instantiated negated goal to run but would produce the wrong answer if any variable was bound.) Thus we must have some way of insuring that every negated literal will be bound. In $GUMS_1$ we have used a simple variable typing scheme to achieve this, as will be discussed later.

We have used a variant of the completed database approach to show that a predicate within the scope of a negation is closed. A predicate is closed if and only if it is defined by an iff statement and every other predicate in the definition of this predicate is closed. We allow a metalevel statement *completed(P)* that is used to signify that by predicate P we really intend the iff definition associated with P. This same technique was used by Kowalski [8] to indicate completion. By default we believe *competed(P)* where not indicated. So if *P* is not explicitly closed ***not P*** is decided by default.

Thus in $GUMS_1$ we have the ability to express that a default should be taken from the lack of certain information (i.e. negation as failure) as well as from the presence of certain information (i.e. default rules). For example, we can have a default rule for the programmer stereotype that can conclude knowledge about linkers from knowledge about compilers, as in:

```
default(knows(linkers) if knows(compilers))
```

We can also have a rule that will take the lack of knowledge about compilers as an indication that the user probably knows about interpreters, as in:

```
certain(knows(interpreters)
        if ~ knows(compilers))
```

This system also allows explicit negative facts and default facts. When negation is proved in reference to a negative fact then negation is not considered a default case. Similarly negation as failure is not considered a default when the predicate being negated is closed. Such distinctions are possible because the $GUMS_1$ interpreter is based on a four value logic.

The distinction between truth or falsity by default (i.e. assumption) and truth or falsity by logical implication is an important one to this system. The central predicate of the system is the two argument predicate *show* which relates a goal *G* expressed as a literal to a truth value. Thus **show(Goal,Val)** returns in the variable *Val* the current belief in the literal *Goal*. The variable *Val* can be instantiated to **true**, **false**, **assume(true)**, or **assume(false)**. The meanings of these values are as follows:

true	definitely true according to the current database.
assume(true)	true by assumption (i.e. true by default)
assume(false)	false by assumption
false	definitely not true.

These values represent truth values for a given user with respect to a given stereotype. If the stereotype is not appropriate, then even definte values may have to change.

Having a four value logic allows us to distinguish conclusions made from purely logical information from those dependent on default information. Four value logic also allows a simple type of introspective reasoning that may be useful for modeling the beliefs of the user. We currently use a default rule to represent an uncertain belief about what the user knows or believes, but we could imagine a situation where we would like to model uncertainties that the user has in his beliefs or knowledge. One such predicate is an embeded *show* predicate. For example we might have a rule that a user will use a operating system command that he believe might erase a file only if he is certain that he knows how to use that command. This might encode as:

```
certain(okay_to_use(Command) if
          can_erase_files(Command),
           show(know(Command),true)).
```

Another predicate *assumed(Pred)* will evaluate the truth of *Pred* and "strengthen" the result. That is

```
demo(assumed(P),V) :-
  demo(P,V2),
  strengthen(V2,V).
```

where the strengthen relation maps assumed values into definite values (e.g. **assume(true)** becomes **true**, **assume(false)** becomes **false** and **true** and **false** remain unchanged). The *assumed* predicate is used to express a certain belief from an uncertain knowledge or belief. For example we might want to express a rule that a user will always want to use a screen editor if he believes one may be available.

```
certain(willUse(screenEditor) if
     assumed(available(screenEditor))).
```

The interpreter that $GUMS_1$ is base on is a metalevel interpreter written in Prolog. The interpreter must generate and compare many possible answers to each subquery, because of the multiple value logic and the presence of explicit negative information. Strong answers to a query (i.e. *true* and *false*) are sought first, followed by weak answers (i.e. *assume(true)* and *assume(false)*). Because strong answers have precedence over weak ones, it is not necessary to remove weak information that contradicts strong information.

Another feature of this system is that we can specify the types of arguments to predicates. This type information can be used to allow the system to handle non-ground goals. In our system, a **type** provides a way to enumerate a complete set of possible values subsumed by that type. When the top-level *show* predicate is given a partially instantiated goal to solve, it uses the type information to generate a stream of consistent fully instantiated goals. These ground goals are tried sequentially.

That goals must be fully intantiated follows from the fact that negation as failure is built into the evaluation algorithm. Complex terms will be instantiated to every pattern allowed by the datatype given the full power of unification. To specify the type information, one should specify argument types for a predicate, subtype information and type instance information. For example, the following says that the **canProgram** predicate ranges over instances of the type **person** and **programmingLanguage**, that the type **functionalLanguage** is a sub-type of **programmingLanguage** and

that the value **scheme** is an instance of the type **functionalLanguage**:

```
declare(canProgram(person,
                   programmingLanguage)).

subtype(programmingLanguage,
        functionalLanguage).

inst(functionalLanguage,scheme).
```

Limitations of the Present System

Our current system has several limitations. One problem is that it does not extract all of the available information from a new fact learned of the user. If we assert that a predicate is *closed*, we are saying that the set of (certain) rules for the predicate form a definition, i.e. a neccessary and sufficient description. In our current system, however, the information still only flows direction! For example, suppose that we would like to encode the rule that a user knows about *I/O redirection* if and only of they know about *files* and about *pipes*. Further, let's suppose that the default is that a person in this stereotype does not know about *files* or *pipes*. This can be expresses as:

```
certain(knows(io_redirection) if
          knows(pipes),
          knows(files)).

default(~knows(pipes)).

default(~knows(files))

closed(knows(io_redirection)).
```

If we learn that a particular user does know about *I/O redirection* then it should follow that she neccessarily knows about both *files* and *pipes*. Adding the assertion

```
certain(knows(io_redirection))
```

however, will make no additional changes in the data base. The values of **knows(pipes)** and **knows(files)** will not change! A sample run after this change might be :

```
?- show(knows(io_redirection),Val).
        Val = true

?- show(knows(pipes),Val).
        Val = assume(false)

?- show(knows(files),Val).
        Val = assume(false).
```

The reason for this problem is that the current interpreter was designed to be able to incorporate new information without actually using a full truth maintenance system. Before a fact *F* with truth value *V* is to be added to the data base, *GUMS*$_1$ checks to see if an inconsistent truth value *V'* can be derived for *F*. If one can be, then a new stereotype is sought in which the contradiction goes away. New knowledge that does not force an obvious inconsistency within the database is added as is. Neither redundant information or existing default information effect the correctness of the interpreter. Subtler inconsistencies are possible, of course.

Another limitation of the current system its inefficiency. The use of default rules requires us to continue to search for solutions for a goal until a strong one is found or all solutions have been checked. These two limitations may be addressable by redesigning the system to be based on a forward chaining truth maintenance system. The question is whether the relative efficiency of forward chaining will offset the relative inefficiency of truth maintenance. The use of an assumption based truth maintenance system [3] is another alternative that we will investigate.

The *GUMS*$_1$ Command Language

Our current experimental implementation provides the following commands to the application.

show(Query,Val) succeeds with *Val* as the strongest truth value for the gaol *Query*. A Query is a partially or fully instantiated positive or negative literal. Val is return and is the value the current belief state. If *Query* is partially instantiated then it will return more answers upon backtracking if possible. In general one answer will be provided for every legal ground substitution that agrees with current type declarations.

add(Fact,Status) sets belief in *Fact* to true. If *Fact* or any legal instance of it contradicts the current belief state then the user model adopts successively higher stereotypes in the hierarchy until one is found in which all of the added facts are consistent. If no stereotype is successful then no stereotype is used, all answers will be based entirely on added facts. *Fact* must be partially or fully instantiated and can be either a positive or negative literal. *Status* must be uninstantiated and will be bound to a message describing the result of the addition (e.g. one of several error messages, *ok*, the name of a new stereotype, etc.).

create_user(UserName,Stereotype,File,Status) stores the current user if necessary and creates a new user who then is the current user. *UserName* is instantiated to the desired name. *Stereotype* is the logical name of the stereotype that the system should assume to hold. *File* is the name of the file that information pertaining to the user will be stored. *Status* is instantiated by the system and returns error messages. A user must be created in order for the system to be able to answer queries.

store_current(Status) stores the current users information and clears the workspace for a new user. *Status* is instantiated by the system on an error.

restore_user(User,Status) restores a previous user after saving the current user if necessary. *User* is the name of the user. *Status* is instantiated by the system to pass error messages.

done stores the system state of the user modeling system, saving the current user if necessary. This command should be the last command issued and needs to be issued at the end of every session.

Conclusions

Many interactive systems have a strong need to maintain models of individual users. We have presented a simple architecture for a general user modeling utility which is based on the ideas of a default logic. This approach provides a simple system which can maintain a database of known information about users as well as use rules and facts which are associated with a stereotype which is believed to be appropriate for this user. The stereotype can contain definite facts and define rules of inference as well as default information and rules. The rules can be used to derive new information, both definite and assumed, from the currently believed information about the user.

We believe that this kind of system will prove useful to a wide range of applications. We have implemented an initial version in Prolog and are planning to use it to support the modeling needs of several projects. We are also exploring a more powerful approach to user modeling based on the notion of a *truth maintenance system*.

Bibliography

1. Clark, Keith L. Negation as Failure. In *Logic and Databases*, J. Minker and H. Gallaire, Ed., Plenum Press, New York, 1978.

2. Reiter, R. Closed World Databases. In *Logic and Databases*, H. Gallaire & J. Minker, Ed., Plenum Press, 1978, pp. 149-177.

3. DeKleer, J. An Assumption Based Truth Maintenance System. Proceedings of IJCAI-85, IJCAI, August, 1985.

4. Finin, T.W. Help and Advice in Task Oriented Systems. Proc. 7th Int'l. Joint Conf. on Art. Intelligence, IJCAI, August, 1982.

5. Howe, A. and T. Finin. Using Spreading Activation to Identify Relevant Help. Proceeding of the 1984 Canadian Society for Computational Studies of Intelligence, CSCSI, 1984. also available as Technical Report MS-CIS-84-01, Computer and Information Science, U. of Pennsylvania.

6. Joshi, A., Webber, B. & Weischedel, R. Preventing False Inferences. Proceedings of COLING-84, Stanford CA, July, 1984.

7. Joshi, A., Webber, B. & Weischedel, R. Living Up to Expectations: Computing Expert Responses. Proceedings of AAAI-84, Austin TX, August, 1984.

8. Kowalski, Robert. *Logic for Problem Solving*. North-Holland, New York, 1979.

9. McDermott, D and J. Doyle. "Non-Monotonic Logic I". *Artificial Intelligence 13*, 1-2 (1980), 41 - 72.

10. Motro, A. Query Generalization: A Method for interpreting Null Answers. In Larry Kerschberg, Ed., *Expert Database Systems*, Benjamin/Cummings, Menlo Park CA, 1985.

11. Pollack, M. Information Sought and Information Provided. Proceedings of CHI'85, Assoc. for Computing Machinery (ACM), San Francisco CA, April, 1985, pp. 155-160.

12. Reiter, Ray. "A Logic for Default Reasoning". *Artificial Intelligence 13*, 1 (1980), 81-132.

13. Rich, Elaine. "User Modeling via Stereotypes". *Cognitive Science 3* (1979), 329-354.

14. Schuster, E. and T. Finin. VP2: The Role of User Modelling in Correcting Errors in Second Language Learning. Proc. Conference on Artificial Intelligence and the Simulation of Behavior, AISB, 1985.

15. Shrager, J. and T. Finin. An Expert System that Volunteers Advice. Proc. Second Annual National Conference in Artificial Intelligence, AAAI, August, 1982.

16. Webber, B. and T. Finin. In Response: Next Steps in Natural Language Interaction. In *Artificial Intelligence Applications for Business*, W. Reitman, Ed., Ablex Publ. Co., Norwood NJ, 1984.

Appendix - The Demo Predicate

This appendix defines the *demo* predicate which implements the heart of the $GUMS_1$ interpreter. The relation

`show(Goal,Value)`

holds if the truth value of proposition *Goal* can be shown to be *Value* for a particular ground instance of *Goal*. The *show* predicate first makes sure that *Goal* is a ground instance via a call to the *bindVars* predicate and then invokes the meta-evaluator *demo*. The relation

`demo(Goal,Value,Level)`

requires that *Goal* be a fully instantiated term and *Level* be an integer that represents the level of recursion within the demo predicate. The relation holds if the "strongest" truth value for *Goal* is *Value*.

```
:- op(950,fy,'~').
:- op(1150,xfy,'if').

show(P,V) :- bindVars(P), demo(P,V,0).

% truth values
demo(P,P,_) :- truthValue(P),!.

% reflection...
demo(demo(P,V1),V,D) :-
  !,
  nonvar(V1),
  demo(P,V1,D) -> V=true;V=false.

% disjunction ...
demo((P;Q),V,D) :- !,
  demo(P,V1,D),
  demo(Q,V2,D),
  upperbound(V1,V2,V).

% conjunction ...
demo((P,Q),V,D) :- !,
  demo(P,V1,D),
  demo(Q,V2,D),
  lowerbound(V1,V2,V).

% negation ...
demo(~P,V,D) :- !,
  demo(P,V1,D),
  negate(V1,V,P).

% assumption ...
demo(assumed(P),V,D) :- !,
  demo(P,V1,D),
  strengthen(V1,V).

% call demo1 with deeper depth and then cut.
demo(P,V,Depth) :-
  Deeper is Depth+1,
  demo1(P,V,Deeper),
  retractall(temp(_,Deeper)),
  !.

% definite facts...
demo1(P,true,_) :- certain(P).
demo1(P,false,_) :- certain(~P).

% find a definite rule that yields TRUE or FALSE.
demo1(P,V,D) :-
  forsome(certain(P if Q),(demo(Q,V,D),demoNote(V,D))).

demo1(P,V,D) :-
  forsome(certain(~P if Q),
          (demo(Q,V1,D),
           negate(V1,V,P),
           demoNote(V,D))).

% stop if the best so far was ASSUME(TRUE).
demo1(P,assume(true),D) :-
  retract(temp(assume(true),D)).

% default positive facts.
demo(P,assume(true),_) :- default(P).

% try default rules 'til one gives a positive value.
demo1(P,assume(true),D) :-
  forsome(default(P if Q),(demo(Q,V,D),positive(V))).

% default negative facts.
demo(P,assume(false),_) :- default(~P).

% default negative rules.
```

```
demo1(P,assume(false),D) :-
  forsome(default(~P if Q),(demo(Q,V,D),positive(V))).

% if P is closed, then its false.
demo1(P,false, ) :- closed(P),!.

% the default answer.
demo1(P,assume(false),_).

% demoNote(X,D) succeeds if X is TRUE or FALSE,
% otherwise it fails after updating temp(A,_)
% to be the strongest value known so far.

demoNote(V,_) :- known(V).
demoNote(V,D) :-
  not(temp(_,D)),
  !,
  assert(temp(V,D)),
  fail.
demoNote(assume(true),D) :-
  retract(temp(_,D)),
  !,
  assert(temp(assume(true),D)),
  fail.
```

% Relations on Truth Values

```
positive(X) :- X == true ; X == assume(true).

known(X) :- X == true ; X == false.

higher(true,_).
higher(assume(true),assume(false)).
higher(_,false).

upperbound(X,Y,Z) :- higher(X,Y) -> Z=X ; Z=Y.

lowerbound(X,Y,Z) :- higher(X,Y) -> Z=Y ; Z=X.

strengthen(assume(X),X).
strengthen(true,true).
strengthen(false,false).

% negation is relative to a predicate.
negate(true,false,_).
negate(assume(true),assume(false),_).
negate(assume(false),assume(true),_).
negate(false,true,P) :- closed(P).
negate(false,assume(true),P) :- not(closed(P)).

truthValue(true).
truthValue(false).
truthValue(assume(X)) :- truthValue(X).
```

% The Type System

```
% isSubtype(T1,T2) iff type T1 has an
% ancestor type T2.
isSubtype(T1,T2) :- subtype(T1,T2).
isSubtype(T1,T2) :-
  subtype(T1,T),
  isSubtype(T,T2).

% true if instance I is descendant from type T.
isInstance(I,T) :- inst(I,T).
isInstance(I,T) :-
  isSubtype(T1,T),
  isInstance(I,T1).

% true if T is a type.
isType(T) :- inst(_,T).
isType(T) :- subtype(T,_).
isType(T) :- subtype(_,T).
```

% Grounding Terms

```
% bindVars(P) ensures that all variables
%  in P are bound or it fails.
bindVars(P) :- var(P),!,fail.
bindVars(P) :- atomic(P),!.
bindVars(P) :-
  schema(P,PS),
  P =.. [_|Args],
  PS =.. [_|Types],
  bindArgs(Args,Types).

bindArgs([],[]).
bindArgs([Arg|Args],[Type|Types]) :-
  bindArg(Arg,Type),
  bindArgs(Args,Types).

bindArg(Arg,Type) :-
  var(Arg),
  isInstance(Arg,Type).
bindArg(Arg,_) :- bindVars(Arg).

% scheme(P,S) is true if S is the schema for P, eg
% schema(give(john,X,Y),give(person,person,thing)).

% find a declared schema.
schema(P,S) :-
  functor(P,F,N),
  functor(S,F,N),
  declare(S),
  !.

% use the default schema F(thing,thing,...).
schema(P,S) :-
  functor(P,F,N),
  functor(S,F,N),
  for(I,1,N,arg(I,S,thing)),
  !.
```

An Efficient Tableau-Based Theorem Prover

Franz Oppacher
Ed Suen

School of Computer Science, Carleton University
Ottawa, Ontario, Canada, K1S 5B6

Abstract

A tableau-based theorem prover, HARP (Heuristics-Augmented Refutation Procedure), is presented which is able to solve the original first-order version of Schubert's Steamroller Problem in a time frame comparable to solutions using a many-sorted version of the problem. HARP accepts the entire language of first-order logic (i.e. it does not require conversion of input expressions to any canonical form) and it uses instantiation instead of unification. HARP's control structure is based on Smullyan's method of analytic tableaux, augmented with explicitly represented and thus easily modifiable heuristics. These heuristics enable HARP to construct its proofs in an efficient as well as human-like manner, making it very suitable for interactive theorem proving.

1. Introduction

This paper describes a complete theorem prover for first-order predicate logic that combines the semantic tableau technique due to [1], [7],and [14] with declaratively expressed heuristic strategies. The resulting proof procedure (HARP) is "natural" and easy to use for people (see Section 2) as well as computationally efficient (for example, as will be described in section 5, HARP is the only theorem prover known to the authors that is able to solve the unmodified, nonclausal version of Schubert's Steamroller Problem [18]).

Since HARP is intended to be used both interactively and as an inference engine for AI applications, its construction is influenced by the overall design goals of a) naturalness, b) efficiency, c) usefulness in an AI environment, and d) partial specification of the control structure by heuristic rules.

a) The goal of naturalness dictates the use of the full language of first order logic and a natural style of proof construction. Our algorithm seems to be unique in that it requires no conversion to any canonical form and accepts its input without any preprocessing. For example, Bibel's system [2] presupposes that negations are driven inward until they apply only to atoms. Even nonclausal resolution [10] which is claimed to be least restricted in truth-functional form, requires quantifier free formulae.

A natural style of proof should deploy principles of inference commonly used by people. Therefore we have opted for normal quantifier instantiation rules instead of unification and for natural-deduction-like rules instead of resolution or connection graph methods which are favored in current approaches[9], [2]. We feel that the method of analytic tableaux [14] provides an extremely natural and elegant codification of logic. Even Robinson uses tableaux to introduce the idea of resolution [12]. Many introductory logic texts also rely on tableaux as a didactic medium (eg. [8]). The natural flavor of the tableau calculus is due to the fact that its rules are syntactic reformulations of semantic evaluation rules [15]. It also allows for an extremely lucid semantic adequacy proof [15], [6].Furthermore, the tableau calculus can be easily extended to modal and other nonclassical logics [6].

b) The addition of an indexing scheme and heuristics to the basic control structure of analytic tableaux [14] results in a highly efficient proof procedure. HARP's empirical efficiency is demonstrated by the solution of accepted benchmark problems, in particular Schubert's Steamroller[18] (see Section 5).

c) Since HARP is intended to work in an AI environment, it has to cope with situations in which there are many irrelevant premises and requests to prove nontheorems. An indexing scheme allows HARP to select relevant premises and heuristics are used to recover quickly from many attempts to prove nontheorems. To date little work has been done on the use of heuristics to detect nontheorems. A brief discussion of this issue will be given in section 4.

d) Because of the experimental nature of the use of heuristics in theorem proving, a control structure modifiable by a set of explicitly represented heuristics is preferable to a rigid architecture where the heuristics are hardwired into the control structure. Moreover we feel this modifiable control structure is crucial to our current investigation of strategy - learning theorem provers (eg. [4]). However this issue will not be discussed in this paper.

2. Background

HARP implements a version of the semantic idea that proving a formula amounts to an unsuccessful attempt to construct a falsifying model for it. This approach to proving by model construction was first studied, although not in the context of automated theorem proving, by [1], [7],and [14]. The method provides a comprehensive test for consistency and, hence, for inconsistency, logical truth and logical entailment.

To prove that Φ is logically true, $\neg\Phi$ is assumed to be semantically consistent or satisfiable. Similarly,to prove that $\Phi_1,...,\Phi_n$ imply Φ, the set $\{\Phi_1,...,\Phi_n,\neg\Phi\}$ is assumed to be satisfiable. From the satisfiability assumption smaller and smaller (weak) subformulas are derived. In either case, if the assumption of consistency is substantiated, a counter-example to the claim that Φ is logically true or that $\Phi_1,...,\Phi_n$ imply Φ can be directly retrieved from the constructed model. However, if the assumption of consistency is untenable, then

the method will ascertain this in a finite number of steps. In other words, the method is complete.

It is convenient to represent models by binary trees. The set of sentences $\{\Phi_1,...,\Phi_n,\neg\Phi\}$ to be tested for consistency comprises the root of the tree. The set is consistent if there is at least one truth-value assignment to all the atomic subformulae of $\Phi_1,...,\Phi_n,\neg\Phi$ on which the latter are simultaneously true. The consequences of the assumption that $\{\Phi_1,...,\Phi_n,\neg\Phi\}$ is consistent are then developed systematically, i.e. in a manner that ensures that no opportunity for generating contradictions is overlooked.

The tree is extended by inserting on its branches the results of applying decomposition rules (see below) to sentences previously entered. This process continues until atomic subformulas of $\Phi_1,...,\Phi_n,\neg\Phi$ or negations of such are reached. The repeated application of decomposition rules to a root node labelled with a consistent set generates a tree with at least one consistent branch, i.e. a branch that does not contain both some atomic sentence and its negation. Conversely, if every branch of the tree is inconsistent, the original set is inconsistent as well.

The decomposition rules are similar to natural deduction elimination rules. When such a rule is applied to any compound sentence that is not a universal quantification the compound is checked off to prevent further rule applications to it and its subformula or subformulae are attached to the end of each open, i.e. consistent branch which passes through the compound. As soon as a contradiction turns up on a branch, the latter is said to be closed and marked with a "*".

A tree is said to be closed if all its branches are closed, and is said to be open otherwise. A branch is said to be complete if it contains either no unchecked compounds or the universal quantifications on it have already been instantiated with respect to all individual parameters appearing anywhere on the branch. A tree is said to be complete if all its branches are complete.

Any complete, open branch provides an interpretation under which the sentences at the root are simultaneously satisfied. Such an interpretation may be useful for certain question-answering tasks. Different complete, open branches may provide different interpretations. The interpretation determined by a branch results from assigning the truth-value **T** to the unnegated atomic sentences on the branch and **F** to the negated ones.

HARP also enters dependency information as part of the tree construction process in order to record for each node all the nodes on which that node depends. This dependency information is used for internal system functions (eg. proof condensation, and graphic display of the proof tree). These dependencies are also used in applications (not described in this paper) that involve nonmonotonic forms of reasoning.

In the tree calculus all sentences are classified as either elementary, i.e. atoms or negations of atoms, or as of type α, β, γ or δ. Sentences of type α, the conjunctive type, are those truth-functionally complex sentences from whose truth one can uniquely infer the truth values of their immediate subsentences. Sentences of type β, the disjunctive type, are all the other truth-functionally complex sentences. Sentences of type γ, the universal type, are those quantificationally complex sentences from whose truth one can uniquely infer the truth values of their instantiations. Sentences of type δ, the existential type, are all the other quantificationally complex sentences.

The semantic rationale for this classification lies in the fact - used in the completeness and correctness proofs - that for each truth value assignment *a* the following holds:

a satisfies $\alpha \Leftrightarrow$ *a* satisfies α_1 and *a* satisfies α_2;
a satisfies $\beta \Leftrightarrow$ *a* satisfies β_1 or *a* satisfies β_2;
a satisfies $\gamma \Leftrightarrow$ *a* satisfies $\gamma[\tau]$ for each parameter τ;
a satisfies $\delta \Leftrightarrow$ *a* satisfies $\delta[\tau]$ for some parameter τ.

In the following,the usual predicate-logical definitions of terms and well-formed formulas are assumed. A "|" in a rule indicates a branch in a tree. We shall use "$\forall$" and "$\exists$" for the universal and existential quantifiers,respectively.

The rules are to be understood as follows: from a formula of the structure shown above the line, derive a formula (or formulas) of the structure shown below the line.

Type α:

$$\frac{\neg\neg\Phi}{\Phi} \qquad \frac{\Phi \wedge \Psi}{\begin{array}{c}\Phi\\ \Psi\end{array}} \qquad \frac{\neg(\Phi \vee \Psi)}{\begin{array}{c}\neg\Phi\\ \neg\Psi\end{array}} \qquad \frac{\neg(\Phi \Rightarrow \Psi)}{\begin{array}{c}\Phi\\ \neg\Psi\end{array}}$$

Type β:

$$\frac{\Phi \vee \Psi}{\Phi \mid \Psi} \qquad \frac{\Phi \Rightarrow \Psi}{\neg\Phi \mid \Psi} \qquad \frac{\neg(\Phi \wedge \Psi)}{\neg\Phi \mid \neg\Psi} \qquad \frac{(\Phi \Leftrightarrow \Psi)}{\begin{array}{c}\Phi \mid \neg\Phi\\ \Psi \mid \neg\Psi\end{array}} \qquad \frac{\neg(\Phi \Leftrightarrow \Psi)}{\begin{array}{c}\Phi \mid \neg\Phi\\ \neg\Psi \mid \Psi\end{array}}$$

Type γ:

$$\frac{\forall\zeta\phi(\zeta)}{\phi(\tau)} \qquad \frac{\neg\exists\zeta\phi(\zeta)}{\neg\phi(\tau)}$$

with proviso (1).

Type δ:

$$\frac{\exists\zeta\phi(\zeta)}{\phi(\tau)} \qquad \frac{\neg\forall\zeta\phi(\zeta)}{\neg\phi(\tau)}$$

with proviso (2).

Proviso (1): Let $\phi(\zeta)$ be any schema and ζ any variable which may or may not occur free in $\phi(\zeta)$. Let $\phi(\tau)$ be the result of replacing all free occurrences, if any, of ζ in $\phi(\zeta)$ by the term τ. The rule of universal instantiation permits the inference from $\forall\zeta\phi(\zeta)$ to $\phi(\tau)$, provided that no free occurrence of ζ in $\phi(\zeta)$ is within the scope of an occurrence of $\forall\tau$ or $\exists\tau$. This restriction prevents illegal capturing of the instantiating variable by a quantifier .(For example, this restriction blocks the invalid inference from $\forall y(Hyz \Rightarrow \exists z(Gz \wedge Jyy))$ to $(Hzz \Rightarrow \exists z(Gz \wedge Jzz))$.)

Proviso (2): Let $\phi(\zeta)$ and $\phi(\tau)$ be as before.When an existential quantifier is dropped the instantiating term τ must be an individual parameter new to the entire branch. In the implementation, it is easier to use a constant foreign to the entire tree.

In principle, a universal quantification is instantiated with respect to every individual term anywhere on its branch unless such an instance is already on the branch. In practice, HARP relies on its full indexing scheme and heuristics (see Sections 3,4) to try first only those instantiations which are most likely to contribute to the early closing of a branch. Only if no individual parameter has appeared is a new one chosen for the instantiation. Unlike all other compounds, a universal quantification is not checked off and, thus, can be used

repeatedly.

3. The basic algorithm

The tree calculus described above is implemented with modifications as the basic loop of HARP :

```
Initialization
Loop until proved or nontheorem
    Select a node from the priority queue
    Decompose the node
        - prioritize the descendants using heuristics
    if a branch closure results
    then
        condense the proof
        if there are more branches
        then
            prepare for the next branch
        else
            proved
        endif
    endif
    if current branch is complete, according to heuristics
    then
        nontheorem
    endif
Endloop
```

In the preparatory stage each formula is internally represented as a node. All formulae are put into a completely parenthesized format by making the connective precedences explicit. This enables efficient, O(1) time, node decomposition. All decomposable nodes are kept on a priority queue. The priority of a node is determined via heuristics (see section 4 for examples). These particular heuristics use the information provided by the indexing scheme.

The indexing scheme is implemented by a connection graph-like mechanism. Links are kept between node pairs which consist of corresponding conjugate literals. Conjugate literal links are determined using a matching procedure which is a relaxed form of unification. Universally quantified variables match anything; constants match only the same constants; existentially quantified variables match only the same existentially quantified variables within the same scope. Any literals that are unifiable will also match using our procedure. However since we do not eliminate existential quantifiers, occasionally links will be made between nonunifiable literals.

Each link is labelled with a weight referred to as the link strength. The link strength is currently calculated by a heuristic using the formula $\lceil 32/2^n \rceil$ where n is the combined sum of the β split level of each literal. The β split level of a subformula is the number of β node decompositions necessary to reach the subformula from the top level formula. Since in a dyadic tree branching increases exponentially with the height of the tree, we have decided to express the strength of a link as inversely exponentially related to the number of β node decompositions necessary to reach the closure point specified by that link.

If one of the pair is a γ node then the link will supply a candidate parameter for universal instantiation when our matching procedure matches the universally quantified variable with this parameter. In contrast, [19] colors the link with the most general unifier. The same heuristic is used to calculate the link strength.

> eg. given nodes $\forall x\mathbf{F}x$, $\neg\mathbf{F}b$, $\mathbf{F}a$, the γ node $\forall x\mathbf{F}x$ forms only one link to $\neg\mathbf{F}b$. **b** is also specified as a candidate parameter along with the link strength 32. No link to **Fa** is built because instantiation with **a** would not lead to closure.

The current heuristic which determines the priority of a node simply sums up the link weights of all the links associated with a node.

Decomposition of a node is performed by the rules in section 2. In the special case when an universal formula is decomposed, our universal instantiation procedure is called to produce a constant which is literally substituted (i.e. using the Lisp Subst function) for occurrences of the universally quantified variable. Since the formulae are completely parenthesized, no unnecessary memory allocation or formula duplication is performed.

β node decompositions spawn a new branch of the tree. The ability to return to the state of computation at which the branch was created is made possible by recording the changes made since the most recent branch point. Upon return to this branch point, these changes are undone. Thus instead of saving the entire proof state, only the changes are noted.

If literals are produced by a decomposition, then checking for a possible branch closure is done in O(1) time. Since each literal is treated as a symbol, checking for a closure simply amounts to checking for the existence of the conjugate symbol of each of the produced literals.

Each time a branch is closed, a proof condensation (or redundancy elimination) procedure is applied. Unfortunately, due to lack of space we are unable to describe the procedure in this paper (see instead [11] where we describe this procedure and its special relationship with the heuristics).

4. Heuristics

HARP's basic control structure, with the exception of proof condensation, is a modification of Smullyan's [14] tree calculus and is thus amenable to a similar completeness proof. Although it can be proved that every inconsistent set has a closed tree it is clearly not the case that every tree for an inconsistent set closes. (For instance, a branch might grow forever simply because of the indefinite repeatability of universal instantiation. In order to close a tree for an inconsistent set, the algorithm closes a branch as soon as a contradiction appears on it.)

However,the basic control structure will often build needlessly branching trees. Our heuristics are designed to prune trees without affecting HARP's completeness. (See the remarks below for some qualifications of this claim.) The heuristic rules prioritize tasks, i.e. tasks more likely to lead to early closure are attempted sooner while heuristically less likely closing tasks are only delayed rather than abandoned.

We feel that a knowledge-based approach to heuristics, i.e. treating heuristics as explicitly represented, independently modifiable rules, has several advantages over hardwired heuristics. For example, it is easy to experiment with different control regimes and to study interactions among heuristics; it

is also straightforward to reformulate them for certain applications so as not to reorder but rather to abandon attempted steps, thereby giving up on the goal of completeness in the interests of greater efficiency.

Our heuristics fall into three groups: a) heuristics for efficient and human-like proof construction, b) heuristics for detecting nontheorems, i.e. for discovering as soon as possible when an open branch is complete, and c) domain-specific heuristics.

Below we shall describe a few sample heuristics from groups a) and b). (We do not describe the rules in c) because we have not yet fully investigated them; moreover, no example mentioned in this paper uses them.) No rules in a) affect HARP's completeness property but some rules in b) do. The latter apply in situations in where a set is consistent but the algorithm could never establish that fact conclusively. As an example, suppose we try to prove the invalid formula $\forall x\exists yHxy \Rightarrow \exists y\forall xHxy$. The proof takes the form of the following, potentially infinite branch:

$\forall x\exists yHxy$, $\neg\exists y\forall xHxy$, $\forall y\exists x\neg Hxy$, $\exists yHxy$, Hxa, $\exists x\neg Hxa$, $\neg Hba$, $\exists yHby$, Hbc, etc. Heuristics in b) keep track of this type of loop and declare the above branch complete after a few cycles. This approach will, of course, not work in all cases but - in view of Church's theorem - no theorem prover can detect all nontheorems. It should be noted that successful proofs of theorems do not rely on any heuristics in b), i.e. the latter are only used to detect some types of nontheorems.

Heuristics are expressed in terms of the information provided by the indexing scheme. However, in order to improve readability (and also because of space limitations) we shall present them in English.

H1: Work on a compound until atoms are reached.
H1 implements a depth-first strategy which is appropriate given the goal of closing all branches.

H2: Favor nonbranching rules that introduce a small number of new nodes that have many corresponding conjugate links.
Highly connected α-type nodes where the connections emanate from a small set of literals within the node are preferred. H2 prevents needless tree expansion.

H3: Prefer existential to universal instantiation.
H3 reflects the fact that existential quantifiers, unlike universal ones, are dropped only once. H3 is overridden only when the available indexing information indicates that an universal instantiation would produce a contradiction immediately, as in the following example: Fa, $\forall x\neg Fx$, $\exists yGy$.

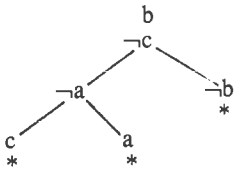
H4: Favor compounds derived from the negation of the conclusion.
H4 corresponds to the set-of-support strategy and aids in selecting relevant premises.

H5: Avoid clearly useless work.
H5 comprises several work reducing rules like the following two: if a propositional variable occurs only once in the premises and not in the conclusion it is thrown away; if no '$\neg$' occurs in the root set then no proof is attempted because at least one branch is sure to be open.

H6: Choose carefully among branching compounds.
According to H6 a conditional is decomposed only if either its antecedent or the negation of its consequent (or both) appear as atoms higher up on the same branch. Similar rules apply to the other β formulae.
The following example illustrates how selective H6 is. Suppose we wish to prove $b \Rightarrow c$, given these premises:

$f \Rightarrow g$, $\neg b \Rightarrow d$, $\neg d \Rightarrow \neg c$, $a \Rightarrow d$, $a \Rightarrow \neg b$, $\neg c \Rightarrow a$.

Then,because of H4 and H6, HARP will build the tree below:



H7: Favor fresh universal quantifications.
For each universal quantifier a record is kept of how often it has been used and with what instantiating terms. In the absence of countervailing heuristic information, less frequently used quantifiers are preferred. H7 reflects the empirical observation that usually in AI applications each relevant premise makes a definite and unique contribution to a proof.

H8: Minimize the introduction of new parameters.
H8 suggests that universal instantiations be done only with respect to terms already on a branch unless there are none yet. Moreover, for the terms on the branch, only those instantiations are made that are sanctioned by the other heuristics.

H9: Identify complete open branches as quickly as possible.
H9 is intended to determine quickly whether the root set of an infinite tree is consistent. If an open branch contains no unchecked compounds except universal quantifications and if every such quantification is instantiated with respect to all and only the terms above it on that branch then the branch may be declared complete. For example, given $\exists xFx$, $\forall x\neg Gx$, Fa, $\neg Ga$, it would be pointless to add further instantiations like $\neg Gb$, $\neg Gc$ etc.

H10: Watch out for nonconverging $\forall\exists$-patterns.
H10, like H9, attempts to identify nontheorems as quickly as possible. H10 consists of several rules from group b) that are invoked whenever a universal quantifier precedes an existential quantifier and universal instantiation is reapplied to new constants introduced by existential instantiation. The various rules of H10 recognize particular $\forall\exists$-patterns that we have empirically identified as frequently indicating infinite branches that will never close. Whenever such a pattern is matched, its branch is immediately declared complete. The basic nonclosing $\forall\exists$-pattern, of which all the other currently detected patterns are variants, is this: $\forall x\exists yFxy$, $\exists yFay$, Fab, $\exists yFby$, Fbc, etc. At this point, H10 decides that it is pointless to continue the branch.

It should be noted that the rules in H10 do not, of course, declare a branch complete as soon as an

$\forall\exists$-pattern is found. Instead, when an $\forall\exists$-pattern is found each rule of H10 begins to look in the current branch for a particular pattern of instantiations that is likely to continue indefinitely. If such a pattern occurs then the branch is declared complete, otherwise these rules have no effect on the proof. As an example, consider the proof of the theorem $\exists x\forall y\ (Fy \Rightarrow Fx)$: $\neg\exists x\forall y\ (Fy \Rightarrow Fx)$, $\forall x\exists y \neg\ (Fy \Rightarrow Fx)$ (at this point, the rules in H10 are activated), $\exists y \neg\ (Fy \Rightarrow Fx)$, $\neg\ (Fa \Rightarrow Fx)$, Fa , $\neg$ Fx, $\exists y \neg\ (Fy \Rightarrow Fa)$, $\neg\ (Fb \Rightarrow Fa)$, Fb, $\neg$ Fa (since the rules have not found a nonconverging pattern of instantiations, the branch closes without their interference).

This concludes our brief description of some sample heuristics.

5. Concluding Remarks

We have presented in this paper an efficient, human-oriented theorem prover as a potential tool in AI environments. To facilitate experimenting with different sets of heuristics - an ongoing concern of ours -, we have pursued the design goal of separating the basic control structure from explicitly stated heuristics.

Analytic tableaux have been used to provide a natural control structure. It appears that the theorem proving community has generally ignored tableaux because of their presumed inefficiency (for a few exceptions see [19], [13]). On the other hand, many logicians find tableaux to form the basis for an elegant and natural codification of logic (e.g [15]).

To make HARP useful as an AI tool, it is designed to quickly detect nontheorems. Also, since dependency information is recorded explicitly in the proof tree, nonmonotonic applications are facilitated.

HARP is currently implemented in approximately 3000 lines of ZetaLisp on a Symbolics 3600. It has solved many of the traditional test problems for theorem provers. These problems include the monkey and banana problem [16], the plane geometry problem [9], problems from [3], and Schubert's Steamroller.

Schubert's Steamroller is a challenge problem that had been unsolved for 6 years until recently in [18]. The quickest solution time of 7.11 secs [18] was achieved by using a many-sorted clausal version of the problem (a further refinement of this approach, a polymorphic many-sorted logic, is described in [5]).

HARP is able to solve the original, nonclausal version of Schubert's Steamroller in 14 seconds on a Symbolics 3600. The resulting proof tree has 48 closure points and is built without any special purpose reasoning mechanisms and without any domain dependent knowledge. It should be noted that the techniques used in converting traditional resolution theorem provers to many-sorted logic are equally applicable to tableaux. For example, the instantiation mechanism could be restricted by using a sort hierarchy of the universe. It is interesting to note that HARP has a solution time comparable to solutions using these techniques. Furthermore HARP seems to be the only theorem prover able to solve the original, nonclausal version of the problem.

Solutions not using a many-sorted version of the problem have also been reported [18]. Without using special knowledge, Stickel's theorem prover has a reported solution time of 2 hours and 53 minutes [5]. The ITP system has a reported solution time of 660 seconds while using a limited many-sorted type logic[18], [5].

Tableaux also appear amenable to parallel architectures. Closing each branch of the proof tree is an independent task. We are currently investigating parallel performance issues using software simulation.

References

[1]: E.W.Beth. *The Foundations of Mathematics.* North-Holland, 1965. 741pp.

[2]: W.Bibel. *"A Comparative Study of Several Proof Procedures ".* Artificial Intelligence (18, 1982), pp 269-293.

[3]: W.W.Bledsoe. *The UT Interactive Prover.* University of Texas at Austin, ATP-17B, 1983.

[4]: D.N.Cohen. *Knowledge Based Theorem Proving and Learning.* UMI Research Press, Ann Arbor, Michigan, 1981. 202pp.

[5]: A.G.Cohn. *"On the solution of Schubert's Steamroller in Many Sorted Logic".* IJCAI 85, pp 1169-1173.

[6]: M.C.Fitting. *Proof Methods for Modal and Intuitionistic Logics.* Reidel, Dordrecht, Holland, 1983.

[7]: J.Hintikka. *Form and Content in Quantification Theory.* Acta Philosophica Fennica 8, 1955.

[8]: R.C.Jeffrey. *Formal Logic: Its Scope and Limits.* McGraw-Hill, New York, 1967.

[9]: D.R.Loveland. *Automated Theorem Proving: A Logical Basis* . North Holland, New York, 1978. 405pp.

[10]: N.V.Murray. *"Completely Non-Clausal Theorem Proving".* Artificial Intelligence (18, 1982), pp 67-85.

[11]: F.Oppacher and E.Suen.*"A Heuristic-Driven, Tableau-Based Theorem Prover",* in preparation

[12]: J.A.Robinson. *Logic: Form and Function.* North Holland, New York, 1981.

[13]: W. Schonfeld. *"Prolog extensions based on Tableau Calculus".* IJCAI 85, pp 730-732.

[14]: R.M.Smullyan.*First-Order Logic,* Springer Verlag, Berlin, 1968.

[15]: W.Stegmuller, M.V.v.Kibed. *Strukturtypen der Logik.* Springer-Verlag, New York, 1984. 524pp.

[16]: C.Walther. *"A many-sorted calculus based on resolution and paramodulation".* IJCAI 83, Vol. 2, pp 882-891.

[17]: C.Walther. *"A Mechanical Solution of Schubert's Steamroller by Many-Sorted Resolution".* AAAI 84, pp 330-334.

[18]: C.Walther. *"A mechanical Solution of Schubert's Steamroller by Many-Sorted Resolution".* Artificial Intelligence, (May 1985), pp 217-224.

[19]: G.Wrightson. *Semantic Tableaux, Unification and Links.* Technical Report CSD-ANZARP-84-001, 21pp, Victoria University, Wellington, New Zealand.

Domain Circumscription Revisited

David W. Etherington[1]
Department of Computer Science
University of British Columbia

Robert E. Mercer[2]
Department of Computer Science
University of Western Ontario

Abstract

Some time ago, McCarthy developed the domain circumscription formalism for closed-world reasoning. Recently, attention has been directed towards other circumscriptive formalisms. The best known of these, predicate and formula circumscription, cannot produce domain-closure axioms; nor does it appear likely that the other forms can. Since these axioms are important in deductive database theory (and elsewhere), and since domain circumscription often *can* conjecture these axioms, we have reason to resurrect domain circumscription. Davis presents an intuitively appealing semantics for domain circumscription. However, under certain conditions McCarthy's syntactic realization of domain circumscription can induce inconsistencies in consistent theories with minimal models. We present a simple, easily-motivated modification which corrects this problem but retains the appealing semantics outlined by Davis. We also explore some of the repercussions of this semantics, including a limited completeness result.

Domain Circumscription

In database and commonsense reasoning, it is often necessary to assume that the only individuals whose existence is relevant to some task are those required to exist by what is known about the task. In such situations, the *domain-closure assumption* is made [9]. This is the assumption that the "world" contains only those individuals whose existence is required by the available information. Reiter [9] observes that this assumption is implicit in relational database theory, where it is entailed by the manner in which universal quantifiers are treated. Thus, for example, in the education database:

$$\begin{array}{ll} Teacher(Smith) & Student(Brown) \\ Teacher(Jones) & Student(Black) \\ Teacher(Plato) & Student(Aristotle) \end{array}$$

with an integrity constraint specifying that the sets of teachers and students are disjoint, even the simple query, "Who are all the teachers?" cannot be answered without implicitly assuming that the domain contains only the listed individuals.

In cases where there are only finitely many individuals, this assumption can be stated using *domain-closure axioms*. These are axioms of the form:

$$\forall x.\ x = t_1 \vee ... \vee x = t_n \tag{1}$$

where the t_i are ground terms. Any model satisfying (1) will have at most n distinct individuals in its domain, those corresponding to the t_i. Reiter [9], [10] shows that domain-closure axioms have an important role in logically formalizing the theory of relational databases.

Domain-closure axioms are also important with respect to a variety of closed-world reasoning formalisms. Perlis and Minker [8], for example, show that the effects of predicate and formula circumscription [6], [7] can be more precisely characterized in conjunction with closed-domain theories. Similarly, Clark [1] requires domain-closure axioms in the development of his predicate-completion approach.

Given the importance of domain-closure axioms, the question arises: Why not explicitly add them to theories? Probably the most important reason is that the appropriate domain-closure axiom may not be obvious. The repercussions of choosing too strong or too weak an axiom (inconsistency or loss of useful conjectures, respectively) argues in favour of a more automatic approach. Furthermore, as the state of the world (or the system's knowledge) changes to bring more entities into consideration, the same mechanism could be used to generate new domain-closure axioms. In certain cases, domain circumscription provides such an automatic mechanism.

A Revised Domain Circumscription Axiom Schema

Domain circumscription [2], [5], [6] is intended to be a syntactic realization of the model-theoretic domain-closure assumption. It provides a mechanism for conjecturing domain-closure axioms, eliminating the need to explicitly state them. To circumscribe the domain of a sentence, A, McCarthy proposes adding the schema:

$$Axiom(\Phi) \wedge A^{\Phi} \supset \forall x.\ \Phi(x) \tag{2}$$

to A. $Axiom(\Phi)$ is the conjunction of $\Phi\alpha$ for each constant symbol α and $\forall x_1...x_n.\ [\Phi x_1 \wedge ... \wedge \Phi x_n] \supset \Phi f x_1...x_n$ for each n-ary function symbol f. A^{Φ} is the result of rewriting A, replacing each universal or existential quantifier, '$\forall x.$' or '$\exists x.$', in A with '$\forall x.\Phi x \supset$ ' or '$\exists x.\Phi x \wedge$ ', respectively.

This axiom schema represents the conjecture that the domain of discourse is no larger than it must be given the sentence A. For any predicate, Φ, if Φ is true for all individuals whose existence is given by the constant terms, through function application, or by existential quantification, and if all individuals in Φ's extension satisfy all of the universally quantified formulae, then Φ can be assumed to contain the entire domain. If the extension of some predicate meeting these requirements is known, it is conjectured to provide an upper-bound on the size of the domain. If the smallest such predicate is known, then the domain is (assumed to be) completely known.

[1] This research was supported in part by NSERC and I.W. Killam predoctoral scholarships, and by NSERC grant 67-7642 to Raymond Reiter.

[2] This research was supported in part by The University of Western Ontario Dean's Research Grant.

The semantic intuition underlying domain circumscription is *minimal entailment*: only those models with minimal domains should be considered in determining the consequences of the given information. In this connection, a model, M, of a sentence is said to be a *submodel* of another model, N, if M is the restriction of N to a subset of N's domain. A model is said to be *minimal* if it has no proper submodels. This notion of submodel corresponds roughly to the standard mathematical notion of 'substructure', although it is slightly stronger.

Davis [2] shows that every instance of (2) is true in all minimal models of the original sentence A. This result is correct for most theories. However, inconsistency results when circumscribing universal theories (theories whose prenex normal forms contain no leading existential quantifiers) with no constant symbols. For example, consider the relational theory:

$$A = \{ \forall x.\ Px \}.$$

Because there are no constant or function symbols, $Axiom(\Phi)$ is empty, so the domain circumscription schema for A is:

$$[\forall x.\ \Phi x \supset Px] \supset \forall x.\ \Phi x\ .$$

Substituting $\neg Px$ for Φx gives:

$$[\forall x.\ Px] \supset \forall x.\ \neg Px\ ,$$

which is clearly inconsistent with A.

The root of this problem is that, for such theories, Φ can be chosen to be universally false. Models of first-order theories must have at least one domain element, so the conjecture that everything is a Φ (and hence there is nothing) is inconsistent. Having isolated the problem, we have developed a simple, easily motivated solution. Since models must have non-empty domains, those Φ's which are identically false must be excluded. To achieve this, the conjunct $\exists x.\ \Phi x$ is added to the left-hand-side of the circumscription schema (2), giving:

$$\exists x.\ \Phi x \wedge Axiom(\Phi) \wedge A^{\Phi} \supset \forall x.\ \Phi(x)\ . \qquad (3)$$

Davis' proof is easily corrected and ammended to apply to this revised schema. Schemas (2) and (3) are equivalent in all but the problematic cases outlined above. If A contains a constant symbol, α, then $\Phi\alpha$ occurs on the left of (2), and this entails $\exists x.\ \Phi x$. Similarly, if A has any leading existential quantifiers, then $\exists x.\ \Phi x$ already occurs in (2). In those cases where $\exists x.\ \Phi x$ is not entailed by the left-hand-side of (2), (2) results in inconsistency. The revised schema may still take a consistent theory – with no minimal models – to an inconsistent circumscription (for an example, see [2]), but so long as A has a minimal model, (3) preserves consistency.

Some Properties of Domain Circumscription

In this section we consider some properties of domain circumscription. We examine their consequences with respect to using domain circumscription to formalize the domain-closure assumption. (Proofs of the results cited here can be found in [3].) For concreteness, we refer to the following example.

Example

Let $T = \{Pa, Pc, Qb, Qc\}$. T has the following minimal models. (We use the corresponding boldface letter for the interpretations of constant terms and $|\ |$ for domains of models and extensions of predicates; α, β, and γ represent the equivalence classes {a, c}, {b, c}, and {a, b, c}, respectively.)

M_1: $|M_1| = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$
$|P|_{M_1} = \{\mathbf{a}, \mathbf{c}\}$
$|Q|_{M_1} = \{\mathbf{b}, \mathbf{c}\}$
$|=|_{M_1} = \{(\mathbf{a},\mathbf{a}), (\mathbf{b},\mathbf{b}), (\mathbf{c},\mathbf{c})\}$

M_2: $|M_2| = \{\alpha, \mathbf{b}\}$
$|P|_{M_2} = \{\alpha\}$
$|Q|_{M_2} = \{\mathbf{b}, \alpha\}$
$|=|_{M_2} = \{(\mathbf{a},\mathbf{a}), (\mathbf{b},\mathbf{b}), (\mathbf{c},\mathbf{c}), (\mathbf{a},\mathbf{c}), (\mathbf{c},\mathbf{a})\}$

M_3: $|M_3| = \{\mathbf{a}, \beta\}$
$|P|_{M_3} = \{\mathbf{a}, \beta\}$
$|Q|_{M_3} = \{\beta\}$
$|=|_{M_3} = \{(\mathbf{a},\mathbf{a}), (\mathbf{b},\mathbf{b}), (\mathbf{c},\mathbf{c}), (\mathbf{b},\mathbf{c}), (\mathbf{c},\mathbf{b})\}$

M_4: $|M_4| = \{\gamma\}$
$|P|_{M_4} = \{\gamma\}$
$|Q|_{M_4} = \{\gamma\}$
$|=|_{M_4} = \{(\mathbf{a},\mathbf{a}), (\mathbf{b},\mathbf{b}), (\mathbf{c},\mathbf{c}), (\mathbf{a},\mathbf{b}), (\mathbf{b},\mathbf{a}), (\mathbf{a},\mathbf{c}), (\mathbf{c},\mathbf{a}), (\mathbf{b},\mathbf{c}), (\mathbf{c},\mathbf{b})\}$

Several important features are evident in the above example. First, every model of T has one of M_1 – M_4 as a minimal submodel. This is not always true, but the class of theories for which it holds is important. It is for such theories that domain circumscription corresponds most closely with one's intuitions. It is clear that theories with only finite models form such a class. It can be shown using a standard result from model-theory that universal theories also have this property.

Second, because the domain circumscription schema is satisfied by every minimal model, domain circumscription does not produce any new ground term equalities or inequalities, at least for the class of theories discussed above. (The same limitation also applies to several other forms of circumscription.) The automatic generation of all possible ground-term inequalities to capture the unique-names assumption [9] remains a thorny issue in knowledge representation.

Third, the ambiguity of the usual statement of the domain-closure assumption is revealed. Only M_4 has the minimum *number* of individuals necessary to satisfy T (*i.e.*, 1), yet each of M_1 — M_4 has only individuals named (and hence required to exist) by T. Domain circumscription captures a weak sense of the domain-closure assumption which does not decide between these interpretations. Based on common applications of the domain-closure assumption (typically in conjunction with some form of unique-names assumption), this weak sense appears to be the preferred sense.

While new ground equality statements are not generally forthcoming, the results of domain circumscription do interact with the equality theory in interesting ways. The circumscription of T in the example above entails $a = b \wedge b = c \supset \exists x \forall y.\ x = y$, for example. The circumscription of $T' = \{\exists x.\ Px,\ \exists x.\ Qx\}$ entails $\exists x.\ Px \wedge Qx \supset \exists x \forall y.\ x = y$. Such formulae seem to precisely capture the difference between the various minimal models of the original theory.

In fact, a completeness result for domain circumscription can be obtained. This result guarantees that, for theories with only finite models (among others), the set of minimal models of the original theory constitutes exactly the set of models of the circumscribed theory. Such a precise characterization is very encouraging.

Related Formalisms

McCarthy [6] claims that domain circumscription is a special case of predicate circumscription, in that the domain circumscription schema for a theory, A, can be derived by predicate circumscription of a theory, A', which is a conservative extension of A. In view of this, it might appear that interest in domain circumscription is pointless. Apart from the fact that domain circumscription is a more direct and somewhat simpler approach to domain-closure, and that the model-theory of domain circumscription perhaps better captures our intuitions about the conjectures involved, there is another reason to reject this argument for abandonment. McCarthy's demonstration of this subsumption actually rests on a strengthened form of predicate circumscription which allows axioms of the original theory to be ignored during the circumscription process. This form of circumscription does not always preserve consistency, even for theories with minimal models. Ordinary predicate circumscription cannot, in general, yield the domain circumscription schema. In fact, this is fortunate, since the form of domain circumscription McCarthy was trying to emulate introduced inconsistencies into some theories with minimal models.

Our revised form of domain circumscription, which preserves consistency for minimally modelable theories, is still not obtainable using predicate circumscription. Etherington, Mercer & Reiter [4] have shown that predicate circumscription is too weak to conjecture domain-closure axioms. Since domain circumscription can conjecture such axioms, it follows that it is not subsumed by its predicate cousin.

McCarthy [7] has outlined a more powerful form of circumscription, called "formula circumscription", which is not subject to some of the limitations of predicate circumscription. Etherington [3] shows that this new approach also fails to subsume domain circumscription. It appears, therefore, that domain circumscription continues to fill a niche among the various mechanisms for closed-world reasoning.

Conclusions

We have pointed out a problem with the version of domain circumscription presented by McCarthy [5], [6]. After isolating the problem, we outlined a straightforward correction which preserves the appealing semantic characterization presented by Davis [2]. Additionally, we have noted the ambiguity of the domain-closure assumption, as it is usually stated. We argue that the most common disambiguation agrees with the results obtained from domain circumscription.

A number of the repercussions of the minimal-model semantics of domain circumscription were explored. Also, we claimed that domain circumscription is complete for certain classes of theories. Finally, we considered the relationship between domain and other forms of circumscription and argued that the connection is not so strong as to justify abandoning domain circumscription.

References

[1] Clark, K.L., "Negation as Failure", in *Logic and Databases*, H. Gallaire and J. Minker (eds.), Plenum Press, New York, 1978.

[2] Davis, M., "The Mathematics of Non-Monotonic Reasoning", *Artificial Intelligence 13*, North-Holland, pp 73-80, 1980.

[3] Etherington, D.W., *Reasoning With Incomplete Information: Investigations of Non-Monotonic Reasoning*, Ph.D. thesis, Department of Computer Science, University of British Columbia, forthcoming, 1986.

[4] Etherington, D.W., Mercer, R.E., and Reiter, R., "On the adequacy of predicate circumscription for closed-world reasoning", *Computational Intelligence 1(1)*, February 1985.

[5] McCarthy, J., "Epistemological Problems of Artificial Intelligence", *Proc. Fifth International Joint Conference on Artificial Intelligence*. 1977, pp 1038-1044.

[6] McCarthy, J., "Circumscription – a Form of Non-Monotonic Reasoning", *Artificial Intelligence 13*, North-Holland, 1980, pp 27-39.

[7] McCarthy, J., "Applications of Circumscription to Formalizing Commonsense Knowledge", *Artificial Intelligence 28*, pp 89-116, 1986.

[8] Perlis, D. and Minker, J., "Completeness Results for Circumscription", *Artificial Intelligence 28*, pp 29-42, 1986.

[9] Reiter, R., "Equality and Domain Closure in First-Order Data Bases", *JACM 27(2)*, 1980, pp 235-249.

[10] Reiter, R., "Towards a Logical Reconstruction of Relational Database Theory", in M.L. Brodie, J. Mylopoulous, and J.W. Schmidt (eds): *On Conceptual Modelling*, Springer-Verlag, New York, pp 191-233, 1986.

TWO TYPES OF QUANTIFIER SCOPING

Sven O. Hurum and Lenhart K. Schubert

Department of Computing Science
University of Alberta
Edmonton, Alberta T6G 2H1

Abstract -- In this paper we argue that a distinction should be made between the scoping of two classes of quantifiers: on the one hand definite and existential quantifiers, which readily escape from strong clausal traps known as "scope islands", and on the other hand distributive quantifiers. We first present evidence that the ability of definite and indefinite NPs to escape from scope islands cannot be accounted for simply by making a "referential/quantificational" distinction. The determination of the scope of these NPs is shown to be closely related to the possibility of regarding them as anaphorically connected. Plural definite and indefinite NPs are interpreted as collections with optional universal partitives. We also describe some preliminary work on the design of a scoping algorithm which we are developing as part of a general purpose natural language understanding system. The algorithm uses a variety of heuristics to compute the relative scoping of pairs of quantifiers (and other logical operators) and combines these to give a list of valid readings in order of preference. We describe some apparent advantages of the algorithm, including the possibilities it allows for the handling wide-scoping definite and indefinite NPs.

1. Introduction

Natural language quantifiers interact with each other and with other logical operators to give rise to a number of types of ambiguity. Such ambiguities are commonly represented in terms of the relative scopes of the operators. The problem of quantifier scoping is then (a) to choose a suitable means of representing scope ambiguities, and (b) to determine a set of constraints and heuristics which can be used to select preferred readings.

A simple ambiguity involving two quantifiers is shown in (1), followed by the two standard interpretations in an informal first order predicate logic (FOPL) with restrictions on quantifiers. In this paper we will use "quantifier" to stand for "quantificational NP".

(1) Everyone loves someone
(2) (Vx:person (Ey:person [x loves y]))
(3) (Ey:person (Vx:person [x loves y]))[1]

We should mention that these glosses are not entirely uncontroversial. Some would argue that (3) is logically redundant since it entails (2), and should therefore be accounted for by pragmatics [1], while others would add a third reading obtained by interpreting *someone* as a "referential" term [2].

[1] Note that we are using an infix sentential syntax in which the predicate follows its first argument. By contrast, functions will be represented in "LISP" prefix notation.

The interaction between two plural quantifiers introduces a further ambiguity which we will represent in an ad hoc way by extending the FOPL to contain indexed quantifiers E_2x:P, E_3x:P, ... and operators *two, three, ...* which operate on ordinary predicates to form predicates over "collections". (Actually, we take an indexed quantifier such as (E_2x:P...) to be equivalent to (Eu: (two P) (Vx: λy[y ∈ u]...)) cf. section 2(c)). We then can distinguish three readings of (4).

(4) Five men painted two walls
(5) (E_5x:man (E_2y:wall [x painted y]))
(6) (E_2y:wall (E_5x:man [x painted y]))
(7) (Eu:(five man) (Ev:(two wall) [u painted v]))

The collective reading (7) indicates that the painting involves one set of men and one of walls, but does not stipulate the fine detail of the relation. Other readings have been described for sentences of this type, such as the "complete group" reading [1], in which each of the men independently painted each of the walls. However, we prefer a non-committal approach to scoping where possible and feel that such readings should be derived from (7) on the basis of meaning postulates (particularly about what it means to perform the relevant actions, or possess the relevant properties, collectively) and pragmatics.

Quantifiers exhibit scope ambiguities relative to verb negation and coordinators. Example (8) may mean that there is a book that Sally hasn't read (9) or that she has read no books (10). The two readings of (11) should be clear.

(8) Sally hasn't read a book
(9) (Ex:book ¬[Sally has-read x])
(10) ¬(Ex:book [Sally has-read x])

(11) Everyone likes Pam or Betty
(12) (Vx:person [x likes Pam] v [x likes Betty])
(13) (Vx:person [x likes Pam]) v
(Vx:person [x likes Betty])

The interaction of quantifiers with opaque operators is more controversial and arguments have been put forward that the standard scope treatment may not adequately capture the different types of ambiguities [3,4]. Nevertheless, it is natural to try to account for ambiguities involving opaque operators in terms of quantifier scope, since the number of readings typically correlates with the number of opaque contexts embedding a quantifier [3]. Opacity is defined by the failure of the "Substitutivity of Identicals". For example, the following argument may be invalid:

(14) John wants to meet the owner of the store
(15) Bill is the owner of the store
(16) John wants to meet Bill

The two standard readings of a sentence containing an indefinite inside an opaque context (17) are shown in (18) and (19).

(17) John wants to marry a blonde
(18) [John wants (Ex:blonde [John marry x])]
(19) (Ex:blonde [John wants [John marry x]])

In the second reading we say that John holds an attitude towards a particular individual whereas in the first he need only have a description "in mind". In this interpretation we follow [2,3]. Widening the scope of the quantifier makes it "specific" relative to the opaque context it scopes outside.

More recently it has been claimed that a similar ambiguity holds for definites and indefinites in transparent contexts [2,5]. For (17) the additional reading would correspond to *the speaker* having a particular blonde in mind. In this case *a blonde* is said to have a "referential" interpretation. However, attempting to account for the tacit attitudes of the speaker seems to us to go beyond the semantic analysis of an utterance as such, ie., it goes beyond the analysis of its "face value" meaning.

There are some further types of scope ambiguity which will not be treated here. Quantifiers may scope with adverbs as in (20) and (21).

(20) John quickly lit all the lamps
(21) Someone always paddles by here on Sundays

(20) may mean that John lit each lamp quickly or that the process of lighting all the lamps was fast. (21) may refer to different persons or to the same person each Sunday. The close parallels between quantifiers and frequency adverbs (some/sometimes, all/always, etc.) makes a unified account of both seem desirable. An attempt to implement such an approach is described in [6].

2. The Scoping of Definites and Indefinites

In this section we will describe some special scoping properties of definite and indefinite NPs which lead us to distinguish between two different types of scoping. We will classify indefinites as in [2] to include, in addition to *a* and *some*, numerals, *a few*, and perhaps *several* and *many* since these share the collective scoping properties considered in this section. In constrast with these are "pure" quantifiers such as *each, every, most, few* and *no*.

We first review the behaviour of indefinites and definites with respect to scope islands, as discussed by Fodor and Sag. In the second subsection we present some counterexamples to their account, and in the third we present our own proposed account.

(a) Scope Islands

All clauses tend to act as "traps" for quantifiers, that is, quantifiers can generally not widen scope over an embedding clause. Some clauses form relatively weak traps, for example clauses which act as verb complements. In exceptional cases, a quantifier such as *each* may widen scope over the subject of an embedding clause thereby creating a functional dependency. Example (22) was used by Vanlehn[7] in an empirical study.

(22) A quick test confirmed that each drug was psychoactive
(23) (Ex:test [x confirmed (Vy:drug [y psychoactive])])
(24) (Vy:drug (Ex:test [x confirmed [y psychoactive]]))

He found that about half the time his subjects chose the second reading, in which there was a different test per drug. Some subjects found the sentence ambiguous.

Certain types of clauses, for example object complements and initial-*if* clauses, form particularly effective traps and these are said to act as "scope islands" (see below). However, indefinites and also definites, if they are treated as scoped elements, show immunity to the scope island constraint.

An extensive study of the scoping properties of indefinites has been made by Fodor and Sag[2]. They claim that indefinites which are immune to the scope island constraint never depend on quantifiers or other logical operators. They use this empirical claim as support for their proposal to make a "referential/quantificational" distinction for indefinites. The behaviour of referential indefinites is to be formalized not in terms of quantifier scope but in terms of an appropriate indexical semantics for Er, a referential variable-binding operator regarded as one of the readings of the article "a(n)". They further suggest that a parallel semantic ambiguity holds for definites.

If correct, this claim would be very important for the development of a scoping algorithm since quantifier movement would then be largely clausebound. However, we feel this claim is flawed. We will first present evidence that indefinites inside scope islands can scope to intermediate positions. The two key sentences used by Fodor and Sag to make their claim are reproduced here as (25) and (29).

(25) Each teacher overheard the rumour that a student of mine had been called before the dean

(26) (Vx:teacher [x overheard the rumour that
(Ey:student of mine
[y had been called before the dean])])
(27) (Vx:teacher (Ey:student of mine
[x overheard the rumour that
[y had been called before the dean]]))
(28) (Ey:student of mine (Vx:teacher
[x overheard the rumour that
[y had been called before the dean]]))

(29) If a student in the syntax class cheats on the exam, every professor will be fired

For the first sentence, they claim that the intermediate reading (27), in which each teacher has a specific student in mind, is not obtained. While this may not be completely convincing (as the authors admit) the absence of an intermediate reading in (29), in which there is a specific student per professor, is very clear.

(b) Specificity and Implicit Anaphora

We feel that the intermediate reading in (25) is hard to get for pragmatic reasons: there is no apparent relation between the teacher and student. If such a relation is present, especially if an explicit anaphor is present, the intermediate reading is easy to get (30). We have replaced *the rumour* with *a rumour* to make the point clearer.

(30) Each teacher overheard a rumour that a favourite student (of his) had been called before the dean

It is also frequently possible to obtain intermediate readings in sentences containing scope islands by embedding such sentences inside a prepositional sentential adverbial (31).

(31) At each school, several students overheard a rumour that a certain teacher had quit

In this case, the "anaphoric relation" between *school* and *teacher* can easily be left implicit. By "implicit anaphora" we mean that certain descriptions may be implicitly understood to contain anaphoric pronouns. This is commonly observed with definite descriptions, as in (32)

(32) Each person asked about the previous owner

where *owner* is understood to mean *owner of it prior to him or her*. There is evidence that some nouns such as *mayor, driver* and *owner* should be treated as binary predicates at the syntactic level and at the level of logical form. In such cases, a "slot" for an implicit anaphor is automatically provided by the logical form translation.

We think the absence of an intermediate reading in (29) can be explained in a different way. We suggest that the consequent clause acts as a fairly strong scope island. The entrapment of *each teacher* inside the consequent clause would then prevent the intermediate reading from being obtained. As evidence for this, we do not get the reading (33), in which the universal quantifier scopes outside the sentence, from (29). The possibility of obtaining an intermediate reading by reversing the order of the clauses (34) appears to result from the universal quantifier being able to escape from the consequent clause in this case.

(33) For each professor, if a student cheats then he will be fired
(34) Each professor may be fired if a (certain) student (he knows) cheats

As further evidence for intermediate readings, we can again embed variations of (29) inside a sentential adverbial or an attitude clause.

(35) In each department, if a (certain) student cheats then every professor will be fired
(36) Each teacher knows that if a (certain) student (of his) cheats he will be reprimanded

It is clearly possible to get specific readings of intermediate scope in (35) and (36). (Modifiers such as *certain* and *particular* are commonly used with wide-scoping, or "specific", indefinites.) We feel that in general indefinites inside scope islands can escape to intermediate positions, provided that some sort of anaphoric relation is apparent.

(c) Interpretation of Definites and Indefinites

When plural definites or indefinites escape from scope islands they cannot create any functional dependencies. For example, in

(37) Someone overheard a rumour that those three races had been cancelled

it is not possible to get a reading in which *someone* functionally depends on the particular race under consideration even though we would like to give *those three races* maximally wide scope. In general, definites and indefinites seldom act distributively when widening scope, even when from the object position within a clause, unless they are preceded by an explicit *each of* partitive.

We can account for these scoping properties by interpreting plural definites and indefinites as ι-(*iota*-) or E-quantified "collections" with optional universal partitives. We assume that the former can readily escape from scope islands whereas the latter cannot. The use of a rule for introducing optional partitives over plural NPs has some independent motivation as it seems necessary for obtaining distributive readings of plural names, generics and indexicals (eg. *we*).

If the initial logical translation of a definite or indefinite plural NP is t (which will denote a collection), the logical form after addition of an optional partitive will be $<V\ \lambda x[x \in t]>$. We use postprocessing rule (38) to obtain this optional partitive

(38) $\quad ti \rightarrow \quad <V\ \lambda x[x \in ti]>, \quad i=1,2, \ldots$

where ti is a variable over terms denoting level-i entities, level-0 entities are ordinary individuals and level-i entities ($i > 0$) are collections of level-(i-1) entities. This allows us to obtain collective and distributive readings without complication in the rules of translation for these NPs.

The following are some examples of our translations for singular and plural indefinites. We translate "a boy" as (a' boy') where a' is lexically ambiguous between the existential quantifier E and an operator μ_1 which forms a "generic instance". The two translations are more accurately represented as $\lambda P<E\ P>$ and $(\mu_1\ P)$ where the angle brackets in the former signify an unscoped quantifier (see next section) while the round brackets signify a functional term. A plural noun such as "boys" is translated as (plur boy') where (plur P) is a predicate true of collections of entities of which P holds. "Two boys" (as a noun) is translated as (two' (plur boy')), where two' modifies (plur boy') so that it will be true only of collections of size two. The bare plural "boys" (as an NP) is translated as $(\mu$ (plur boy')) where μ forms a "kind" or "species" (see [8,9]). The syntactic-semantic rule pairs giving the translations for "a boy" and "boys" are shown in (39)-(41).

(39) NP —> DET N, (DET' N')
(40) NP[PLUR] —> N[PLUR,-NUM], (μ N')
(41) NP[PLUR] —> N[PLUR,NUM], (a' N')

Rule (40) applies only to non-numeral NPs such as "boys", while (41) applies to numeral NPs such as "two boys". The semantic output of rule (41) for "two boys" would be (42), with the two possible lexical disambiguations shown in (43) and (44).

(42) (a' (two' (plur boy')))
(43) <E (two' (plur boy'))>
(44) (μ_1 (two' (plur boy')))

The (43) and (44) translations account for the particular/generic ambiguity in

(45) Two boys can lift this sofa

and the optional distributive rule (38) accounts for the further distributive/collective ambiguity of the particular reading.

In constrast to indefinites and definites, we treat the determiners *most, few* and *every* as being lexically ambiguous between the collective and distributive readings, with a strong preference for the latter. We do this to account for the difficulty in obtaining collective readings with these determiners. As an example, the two translations for *every* are (a) the universal quantifier V and (b) $\lambda P<$the $\lambda x[<V\ P> \in x]>$.

3. Towards a Quantifier Scoping Algorithm

We are in the process of designing a quantifier scoping algorithm which is to be used as an extension to a general purpose natural language understanding system previously described [10]. The parser is based on Generalized Phrase Structure Grammar with semantic rules that generate logical translations in a slightly augmented first order modal logic. The initial logical translations are in general ambiguous, leaving unscoped elements marked (indicated by angled brackets) for further disambiguation. This allows a separate scoping module to be designed which operates on the initial logical form. The eventual intent is that the syntactic, semantic and pragmatic phases should operate in parallel.

As an example, the initial logical form for (46) is shown in (47) and the two scoped readings in (48) and (49). These have been simplified by treating PRES as an identity transformation.

(46) Most people on each committee know John

(47) [<most λx[[x person] & [x on <each committee>]]>
(PRES ly[y know John])]

(48) (most x: [[x person] & (each y: [y committee] [x on y])]
[x know John])
(49) (each y: [y committee]
(most x: [[x person] & [x on y]] [x know John]))

The algorithm being designed is in some respects similar to that described in [11,12] but works "bottom-up", carrying up lists of readings at each level in order of preference. Each reading contains a quantifier, variable, restriction and (possibly) a main predication, preceded by a weighted "preference" value. However, in order to obtain the set of all valid scope orderings, quantifiers which widen scope do not immediately embed the logical forms over which they scope. For example, the two NPs in (50) will return the two lists of scope orderings shown in (51) and (52). [2]

(50) Most people on each committee attended a meeting

(51) ((0.8 (each y [y committee])
(most x [[x person] & [x on y]]))
(0.2 (most x [[x person] &
(each y [y committee] [x on y])])))
(52) ((1.0 (a z [z meeting])))

In (51) the weighting results from the use of two heuristics: (a) the tendency of a quantifier in a prepositional noun complement to scope over the head noun and (b) the lexical tendency of *each* to take wide scope. In this case the two lists will be combined at the clausal level to give five readings.

By keeping the lists sorted it is possible to use cutoff points to minimize the exponential growth of readings. In many cases, semantic or pragmatic knowledge may also be used to disallow certain quantifiers, and two existential readings need not be scoped relative to each other. If all readings are considered at each level, a complete set of the valid readings of a sentence will be generated.

[2] These temporary logical forms will differ somewhat when the algorithm is extended to scope coordinators.

The algorithm makes use of a major structural constraint on scoping which is described in [12] and which follows from the principle of "quantifier raising": a quantifier may not scope between a noun and its complement. Therefore, in (50) *a meeting* may not scope inside *most* but outside *each*. Also, an NP with two nested complements such as "A of (B of C)" cannot have the scope ordering "BAC".

We feel that a "bottom-up" approach to scoping is well worth pursuing. Since it parallels the "bottom-up" parsing process it should make it easier to combine syntactic and semantic heuristics. This would also eventually make it possible to run the parsing and scoping procedures in parallel. As was mentioned, it is possible to use cutoff points to trim the explosive growth of readings. It is also possible (in principle) to store the preference lists at each NP or clause to allow subsequent re-evaluation of certain operator pairs without rerunning the entire algorithm.

Since wide-scoping definites and indefinites are likely to be anaphoric or to take maximally wide scope, they may be tagged with an indicator of their anticipated anaphoric referents. This would allow early elimination of many of the scope ordering possibilities which arise when definites and indefinites widen scope over embedding clauses.

4. Some Heuristics

A good introduction to scoping heuristics is given by Vanlehn[7] and some of the heuristics he describes have been used in recent implementations [12]-[16]. We will briefly describe some of the heuristics we are using here.

(a) Entrapment

As has been mentioned earlier, some clauses form strong scope islands whereas others occasionally permit a distributive quantifier to widen scope over an embedding clause. Also, VPs serving as verb or noun complements or as adverbials generally create weak traps. As an example, Vanlehn has shown that quantifiers in VPs serving as noun complements (reduced relative clauses) have an intermediate likelihood of widening scope over the head quantifier compared to those in prepositional phrases and full relative clauses. We can make use of this heuristic by checking for a *progressive* feature on the VP of the reduced relative clause.

(b) Surface Order

In the absence of other determining factors scope order tends to follow surface order. This heursitic appears to be most useful for the scoping of preposed prepositional adverbials and of subject and object. The subject-over-object preference can be seen in

(53) Few people like everyone
(54) Few people are liked by everyone

in which the *few-everyone* readings are predominant in both cases.

(c) Lexical Heuristics

Vanlehn describes a correlation between the tendency of a quantifier to take wide scope and its tendency to form a distributive as opposed to a collective NP. The most reliable heuristic here is the tendency for *each* to take wide scope. However, as we have discussed, the wide scoping behaviour of definites and indefinites also needs to be taken into account; ie., existentially quantified and definite terms can escape from scope islands where *each* cannot, and this tendency runs counter to the (localized) wide-scoping tendency of *each*.

(d) Negation and Coordination

Verb negation has a strong tendency to scope outside univeral quantifiers, even when the latter are in subject position. The commutativity of *or* with existential quantifiers and of *and* with universal quantifiers simplifies scoping considerably. (However, plural indefinites, when interpreted distributively, do scope with *or*, a fact which we account for by using universal partitives to represent them). There are some interesting parallels between the scoping behaviour of these two commutative pairs. For example, *or* can escape from scope islands to both intermediate and maximally wide scope positions whereas *and* appears to be subject to similar clausal constraints on scoping as are universal quantifiers. This supports the position taken in this paper that the apparent escape of definites and indefinites from scope islands is indeed a scoping phenomenon, rather than the result of a semantic referential/quantificational ambiguity.

Acknowledgements

We wish to thank the members of the Logical Grammar Study Group at the University of Alberta for their comments and suggestions. This work was supported by NSERC Operating Grant A8818.

References

[1] R.M. Kempson and A. Cormack (1981), Ambiguity and quantification, *Linguistics and Philosophy 4*, pp. 259-309.

[2] J.D. Fodor and I.A. Sag (1982), Referential and quantificational indefinites, *Linguistics and Philosophy 5*, pp. 355-398.

[3] J.D.Fodor (1976), The Linguistic Description of Opaque Contexts, Ph.D. Dissertation, Published by Indiana University Linguistics Club, 206pp.

[4] E.Saarinen (1980), Quantifier phrases are (at least) five ways ambiguous in intensional contexts, in F.Heny (ed.), Ambiguities in Intensional Contexts, D.Reidel, pp. 1-45.

[5] H.K. Wettstein (1981), Demonstrative reference and definite descriptions, *Philosophical Studies 40*, pp. 241-257.

[6] M.C.McCord (1981), Focalizers, the scoping problem and semantic interpretation rules in logic grammars, *Proceedings of the International Workshop on Logic Programming for Expert Systems*, Logicon, Woodland Hills, CA.

[7] K. Vanlehn (1978), Determining the Scope of English Quantifiers, MIT Artificial Intelligence Laboratory Technical Report AI-TR-483, 123pp.

[8] F.J.Pelletier and L.K. Schubert (1982), Mass terms, in D. Gabbay and F. Guenther (eds.), Handbook of Philosophical Logic, Reidel, Dordrecht.

[9] L.K. Schubert and J.F. Pelletier, Problems in the representation of the logical form of generics, plurals and mass nouns (In Press).

[10] L.K.Schubert and F.J.Pelletier (1982), From English to logic: context-free computation of 'conventional' logical translation, *American Journal of Computational Linguistics 8*, pp. 26-44.

[11] J.R. Hobbs (1983), An improper treatment of quantification in ordinary English, *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, pp. 57-63.

[12] J.R.Hobbs and S.M.Shieber, An algorithm for generating quantifier scopings, (In preparation)

[13] W.A. Woods (1978), Semantics and quantification in natural language question answering, *Advances in Computers 17*, pp. 1-87.

[14] V.Dahl (1979), Quantification in a three-valued logic for natural language question-answering systems, *Proceedings of the Joint International Conference for Artificial Intelligence*, pp. 182-187.

[15] D.H.D. Warren and F.C.N. Pereira (1982), An efficient easily adaptable system for interpreting natural language queries, *American Journal of Computational Linguistics 8*, pp. 110-119.

[16] P. Saint-Dizier (1985), Handling quantifier scoping ambiguities in a semantic representation of natural language sentences, in V.Dahl and P.Saint-Dizier (eds.), Natural Language Understanding and Logic Programming, North-Holland, pp. 49-63.

A Propositional Logic for Natural Kinds

James P. Delgrande

School of Computing Science,
Simon Fraser University,
Burnaby, B.C.,
Canada V5A 1S6

ABSTRACT

An approach for representing knowledge about natural kinds is presented. This is accomplished by adding to propositional logic a "variable conditional" operator to express relations among kinds and attributes of kinds. Truth conditions for this operator are based on a possible-worlds semantics; a proof theory is provided, and the logic is shown to be sound and complete. Properties of the resultant formal system are argued to correspond to common intuitions concerning natural kinds. Moreover the system is argued to provide a more appropriate basis for representing knowledge about such kinds than other existing approaches.

1. Introduction

"Birds fly" and "ravens are black" seem on the face of it to be reasonable assertions to make about the external world and hence reasonable statements to perhaps incorporate into a knowledge base about the world. Yet of course it isn't the case that *all* birds fly — for example birds with broken wings don't, nor do penguins. This is in contrast with an assertion such as "penguins are birds", where presumably if something is a penguin then it cannot help but be a bird.

The problems of "exception-allowing general statements" such as "birds fly" have been addressed within Artificial Intelligence primarily by means of two approaches: default reasoning and prototype theory. In the first case, "birds fly" is interpreted, roughly, as saying that if an object is known to be a bird, and it is consistent with what is believed that it flys, then conclude that it flys. In the second case, the statement is interpreted, again roughly, as "typically birds fly", and flight is asserted to be only a typical characteristic of birdhood.

In this paper, a third alternative is introduced. Here, "birds fly" is interpreted as "all other things being equal, birds fly", or "*ceteris paribus* birds fly", or "ignoring exceptional conditions, birds fly". This approach is intended to be applicable to terms standing for naturally occurring kinds or *natural kinds*, such as "water", "raven", "lemon", etc. The general idea is that an operator $\Rightarrow$ is introduced into the propositional calculus, where $A \Rightarrow B$ is interpreted as "in the normal course of events, if A then B". This allows the consistent assertion, for example, that

$$Raven \Rightarrow Black$$

along with

$$Raven \wedge Albino \Rightarrow \neg Black.$$

Intuitively, the connection between the antecedent and consequent of these conditionals may be thought of as a relationship in some scientific theory (1). Ravens are black, but only if some set of presumed "additional assumptions" are satisfied: the raven isn't albino, hasn't been painted, isn't in a strong red light, etc.

In extending the semantics of propositional logic to account for $\Rightarrow$ I adopt a possible worlds approach, where $A \Rightarrow B$ is true if B is true in the "least exceptional" worlds where A is true, and $A \supset B$ is true in all less exceptional worlds. Roughly then this says that ravens are black, if we "factor out" all exceptional circumstances such as albinoism, being painted, being featherless, etc. While the actual choice of a metric for "less exceptional than" between possible worlds is, to a large extent, a pragmatic affair, we can specify some bounds on such a metric, and these bounds will constrain the semantics of the $\Rightarrow$ operator. The resulting formal system is a "conditional logic", of a class of logics that have been developed for dealing with counterfactual conditionals, conditional obligation, and other such areas. Thus while the system described here differs in significant respects from other conditional logics described in the literature, nonetheless the semantic theory presented is in much the same spirit as these other logics.

These notions are expanded and amplified in what follows. Following an informal exposition a semantics and proof theory for the formal system is presented. The claim is made that this system provides an appropriate means for representing the properties of, and relations among, natural kinds. In the next-to-last section I consider a first-order formulation of the system. Further details and proofs of theorems are given in [5].

The formal system that I will be primarily concerned with is strictly propositional. The issues I wish to address are best illustrated using a propositional treatment and, moreover, the extension to a first-order account presents no great problem. There is though a minor difficulty in regarding *Raven*, *Black*, etc. as propositions. In this framework the atomic propositions are best understood as making assertions about individuals. Hence a somewhat less informal reading of $Raven \wedge Albino \Rightarrow Black$ is "If something is a raven and that thing is an albino then, all other things being equal, that thing is black". I return to this point in the penultimate section.

1. Natural Kinds, Default Logics, and Prototype Theory

A large and important group of terms is that whose members stand for naturally occurring classes, or *natural kinds*. Examples include such common nouns as "water", "raven", "lemon", etc. It does not include terms that can be analytically defined, such as "bachelor" or "square". Natural kind terms may be characterised as being of explanatory importance, but whose normal distinguishing characteristics are explained by deep-lying mechanisms [14]. A difficulty with such terms is that they have virtually no interesting exceptionless properties. Thus for example, while all birds are presumably animals, it certainly isn't the case that all birds fly, or are feathered, or have two legs.

* This research was supported in part by the National Science and Engineering Research Council of Canada grant A0884 and in part by the Simon Fraser University President's Research Grant 02-4014.

(1) Although this begs a host of questions, another reading of the conditional may be "it is a *scientific law* that if A then B".

One general approach in AI for dealing with such terms is default and non-monotonic reasoning. For example, in Reiter's augmentation of first-order logic [15], "ravens are (typically) black" would be represented by the default rule:

$$\frac{Raven(x) : Black(x)}{Black(x)}.$$

This can be interpreted as "if something can be inferred to be a raven, and if that thing can be consistently assumed to be black, then infer that that thing is black". Related approaches are described in [9], [10], and [11]. A general, generic difficulty with these approaches is that their semantics rests on a notion of consistency with a set of beliefs. Thus, paraphrasing the above, a raven may be believed to be black, if this does not conflict with prior beliefs. However, the relation between ravens and blackness, whatever it may be, is clearly independent of any particular believer. Thus, while default reasoning may be fine for telling us how to consistently *extend* a belief set, if we want to attempt to *represent* the relation between ravens, blackness, albinoism, etc. it seems we need look elsewhere.

A second approach in AI for dealing with natural kind terms is prototype theory [16]. In this case (and assuming a first-order theory) membership in the extension of a term is a graded affair and is a matter of typicality or similarity to a representative member or prototype. Thus one might say that it is more ravenlike to be black than some other colour, or perhaps, a raven is black but only with some given certainty. A difficulty with these approaches is that there is no clear agreement as to what is meant by "certainty", nor how one may combine and work effectively with such certainties [19]. However the problems appear to run deeper than this. Prototype theory seems concerned generally with descriptions of individuals, or predicting properties of individuals. Hence such an approach seems useful for recognising an individual as a raven, given a list of characteristics or, alternatively, predicting the colour of an individual given other information about it. However this isn't what we want. Rather we want to attribute blackness of ravens as following in the normal course of events from the essential conditions(2) of ravenhood. In such a case, notions of typicality and resemblance to a prototype appear too weak to be useful.

2. Natural Kinds and Conditional Logics

Consider yet again the assertion "in the normal course of events, ravens are black". We might write this as

$$Raven \Rightarrow Black,$$

reserving $\supset$ for the standard material conditional. We might also want to assert that, in the normal course of events, albino ravens aren't black:

$$Raven \wedge Albino \Rightarrow \neg Black.$$

In a similar fashion, perhaps there is also some disease X, peculiar to albinos that turns them black again:

$$Raven \wedge Albino \wedge Has\_X \Rightarrow Black.$$

The pattern in the above three statements is clear: as the antecedent is strengthened, the consequent may change to its negation. In standard, classical logics a set of such conditionals is satisfiable only if some of the antecedents are false. Thus the above is satisfiable only if *Raven* or *Albino* is false. This though won't do: obviously we want to recognise the existence of albino ravens. So our eventual treatment of $\Rightarrow$ should allow for such formulas to be satisfiable while having true antecedents.

A second instance where the $\Rightarrow$ operator differs from the material conditional is transitivity. Thus if $A \supset B$ and $B \supset C$ then it cannot be other than $A \supset C$. However this need not be so with natural kinds. For example, penguins are birds, and birds (normally) fly, but penguins don't (normally) fly.

Such patterns of inference and deviations from the classical norm have been recognised by philosophers in other types of reasoning. Best known in this regard is counterfactual reasoning and, in this area, perhaps the central work is that of David Lewis [8]. So for example Lewis gives the following instance of strengthening the antecedent:

> "If Otto had come it would have been a lively party; but if both Otto and Anna had come it would have been a dreary party, but if Waldo had come as well, it would have been lively: but " [8 p 10]

Failure of transitivity of the counterfactual conditional is also easy to show. There are though major differences between counterfactual reasoning and representing knowledge about natural kinds. Most significantly counterfactual reasoning treats conditionals where the antecedent is false, but the conditional as a whole may be true or false. In the case at hand, the antecedents are true, but the corresponding material conditionals may be jointly inconsistent.

Related approaches for reasoning with counterfactuals and reasoning about conditional obligation are described in [12], [18], and [20]; while [3] provides a general discussion. These approaches are all examples of *conditional logics*. The underlying semantic theory for such approaches is typically expressed using a possible worlds formulation. The general idea is that the truth value of a conditional $A \Rightarrow B$, relative to a world, depends on a subset of those worlds in which A is true. In this sense the conditional can be regarded as a necessity operator on B but where the subset of "relevant" worlds depends on A; for this reason the operator $\Rightarrow$ is referred to as a *variably strict* conditional, or simply a *variable* conditional. Thus for example in Lewis's approach, $A \Rightarrow B$ is true if the minimal set of worlds (or "sphere") most like our own that have A true also, for those worlds, have B true, and for no world in the sphere is $A \supset B$ false.

A somewhat different approach is developed by Oster Dahl [4] wherein a modal logic is employed directly for representing information about natural kinds. "Birds fly" is interpreted as either "in all alternative worlds, all birds fly sometimes" or "in all alternative worlds, all birds are such that in all alternative worlds, they fly sometimes". Both alternatives however have undesirable properties. In particular, neither allow the truth value of a consequent to change with a strengthening of the antecedent.

In the next section I propose an appropriate accessibility relation for the variably strict conditional that is to be used to govern natural kinds. From this a formal semantics and proof theory are developed in the following sections.

3. A Modal Basis for Natural Kinds

If the world was more uniform, or less exceptional, than it is, then perhaps ravens, with the exception of albinos, would be black. Albino ravens would of course be white. If the world were yet more uniform, then perhaps there would be no albinos at all, and all ravens would be black. This then is the way statements such as "ravens are black", "albino ravens are white", and so on are informally interpreted. In this section I consider a possible worlds formulation for a logic where the accessibility relation E is to be interpreted so that Ew_1w_2 holds between worlds w_1 and w_2 just when w_2 is at least as uniform, or at least as unexceptional, as w_1. From this $A \Rightarrow B$ is defined to be true at a world just when the least exceptional worlds in which A is true also have B true, and in no less exceptional world is A true and B false. The problem of course is to restrict E so that $\Rightarrow$ will conform to our intuitions concerning natural kinds.

From our informal reading of E, the following should clearly hold:

Reflexive: Eww for all worlds w.

Transitive: If Ew_1w_2 and Ew_2w_3 then Ew_1w_3.

(2) presumably associated with genetic makeup

That is, every world is at least as unexceptional as itself, and the notion of unexceptionalness is transitive. On the other hand we clearly don't want the relation to be symmetric. So if necessity was to be defined in this system, the corresponding modal logic would subsume S4 but not S5.

It would seem that this notion of unexceptionalness should lead to a "relatedness" of accessible worlds in the sense that any two worlds accessible from a given world should have a common, accessible, less exceptional world. This condition is expressed by:

Incestual: If Ew_1w_2 and Ew_1w_3 then there is a w_4 such that Ew_2w_4 and Ew_3w_4.

The difficulty with this condition is that, without further restriction, it would allow the formula $A \Rightarrow B \wedge A \Rightarrow \neg B$ to be satisfiable (once a formal semantics is developed). This obviously isn't quite right, and can be avoided if every two worlds accessible from another, are themselves comparable. Hence we adopt:

Forward Connected: If Ew_1w_2 and Ew_1w_3 then either Ew_2w_3 or Ew_3w_2.

Note that this condition is weaker than the euclidean condition, where the "or" is replaced by "and".

Another plausible condition is that as progressively less exceptional worlds are considered from a given world, that the worlds eventually agree on the truth value of a proposition. That is, we have a "convergence" of accessible worlds towards (although not necessarily to) a simplest world. This is guaranteed by the following condition, together with connectedness.

Terminating: For any world w_1 and proposition p there is a w_2 so that Ew_1w_2 and either for all w_3 where Ew_2w_3, p is true at w_3 or else for all w_3 where Ew_2w_3, p is false at w_3.

With these conditions, the accessibility relation for a given world yields an upper semilattice with that world as the upper bound. If only a finite number of primitive propositions are considered, then there is a unique, simplest world where all true propositions are necessarily true; in this case the accessibility relation from any world yields a lattice. Note however that these conditions do not preclude mutually inaccessible worlds.

The modal logic corresponding to this accessibility relation is investigated in [17] as the logic K3; see also [7]. K3 subsumes S4.3; however it neither subsumes nor is subsumed by S5.

4. A Formal Semantics

The language N I shall consider is formed in the usual manner from a denumerable set of atomic sentences $\mathbf{P} = \{p_0, p_1, \cdots\}$, together with standard connectives $\neg$ and $\supset$ for negation and the material conditional, and a new binary connective $\Rightarrow$ for variable conditionality. Conjunction ($\wedge$), disjunction ($\vee$), and biconditionality ($\equiv$) are introduced by definition

Sentences of N are interpreted in terms of a model structure $M = \langle W, E, P \rangle$ where W is a set, E is a reflexive transitive, forward connected, and terminating binary relation on W, and P is a function from $\mathbf{P}$ to subsets of W. This structure corresponds to a particular instance of a standard Kripkean possible worlds structure where, informally, W is a set of possible worlds, E is the accessibility relation on W, and P maps atomic sentences onto those worlds where the sentence is true Ew_1w_2 is read as "w_2 is at least as unexceptional as w_1" The symbolism $\models^M_w A$ will be used for the truth of a statement A with respect to a model structure M and world w.

More formally we have:

(i) $\models^M_w p_n$ iff $w \in P(p_n)$ for every n.

(ii) $\models^M_w \neg A$ iff not $\models^M_w A$.

(iii) $\models^M_w A \supset B$ iff if $\models^M_w A$ then $\models^M_w B$.

(iv) $\models^M_w A \Rightarrow B$ iff (a) there is a $w_1 \in W$ so that Eww_1 and $\models^M_{w_1} A$ and $\models^M_{w_1} B$ and there is no $w_2 \in W$ so that Ew_1w_2 and $\models^M_{w_2} A$ and $\models^M_{w_2} \neg B$, or (b) for every $w_1 \in W$ where Eww_1, $\models^M_{w_1} \neg A$.

If $\models^M_w A$, we say that A is *true* in the model structure M in world w. We write $\models A$ in the case that A is true at every world in every model structure, and say that A is *valid*. A is *satisfiable* if and only if $\neg A$ is not valid.

A variable conditional $A \Rightarrow B$ is true at a world just in case there is some relatively less exceptional world where both A and B are true, and at no still less exceptional worlds is $A \supset B$ false. As a degenerate case, if A is false at all worlds accessible from w, then $\models^M_w A \Rightarrow B$ is taken as true.

We obtain, first and perhaps most importantly, the result that the following set of sentences is satisfiable:

$$\{A_1 \Rightarrow B,\ A_1 \wedge A_2 \Rightarrow \neg B\}$$

where each of A_1, A_2, and B are true. So we can say, without inconsistency, that ravens are (normally) black, but albino ravens are (normally) not black. The pattern of strengthening the antecedent to reverse the truth value of the consequent may clearly be extended arbitrarily. In addition,

$$\{A \Rightarrow B,\ B \Rightarrow C,\ \neg(A \Rightarrow C)\} \text{ and}$$

$$\{A \supset B,\ B \Rightarrow C,\ \neg(A \Rightarrow C)\}$$

are satisfiable and so, as required, we lose transitivity of the conditional. Hence we can say, without inconsistency, that penguins are birds, birds (normally) fly, but it isn't the case that penguins (normally) fly. On the other hand,

$$(A \Rightarrow B) \supset ((B \supset C) \supset (A \Rightarrow C))$$

is valid. So we can also say "ravens are (normally) black; black things are not white; hence ravens are (normally) not white".

Furthermore, we have the fact that while

$$\neg(\neg(A \Rightarrow B) \supset (A \Rightarrow \neg B))$$

is satisfiable,

$$(A \Rightarrow B) \supset \neg(A \Rightarrow \neg B)$$

is valid. What this means is that we do not have a standard law of the excluded middle. Hence if a kind does not (normally) have a particular attribute, we are not bound to attribute to it the negation. Thus perhaps, neither having nor not having a sense of humour arises in the normal course of events from being a penguin. On the other hand, if $A \Rightarrow B$ is satisfiable then $A \Rightarrow \neg B$ is unsatisfiable.

A further property of the semantics is that there are models where both $\models^M_{w_1} A \Rightarrow B$ and $\models^M_{w_2} A \Rightarrow \neg B$ are true. This is in conformity with the intuition that while, for example, ravens are (normally) black, things might have been such that, without altering the essential nature of ravens, these birds (normally) are red. However the above two statements can be true only if w_1 and w_2 are mutually inaccessible. Thus ravens might have been (normally) red, even under the criterion represented by E, but not in any world accessible to ours with respect to E.

Arguably then the connective $\Rightarrow$ does conform to some of our primary intuitions concerning natural kinds. In the following section, the properties of N are further explored by considering the proof theory of the system

5. Proof Theory

Theorems of classical propositional calculus PC must clearly be theorems of N. Thus we can adopt any of the usual sets of axioms for PC together with *modus ponens*. What remains is to characterise the variable conditional $\Rightarrow$. The following axioms and rule of inference are adequate for this purpose.

Id	$A \Rightarrow A$
CC	$(A \Rightarrow B \wedge A \Rightarrow C) \supset (A \Rightarrow B \wedge C)$
RT	$A \Rightarrow B \supset (A \wedge B \Rightarrow C \supset A \Rightarrow C)$

NC $\neg(A \Rightarrow B) \supset (A \Rightarrow C \supset A \wedge \neg B \Rightarrow C)$

CC' $(A \Rightarrow C \wedge B \Rightarrow C) \supset (A \vee B \Rightarrow C)$

ND $(A \vee B \Rightarrow C) \supset (A \Rightarrow C \vee B \Rightarrow C)$

RCM If $B \supset C$ then infer $A \Rightarrow B \supset A \Rightarrow C$

For naming the axioms and rules of inference, I have followed, and will follow, the conventions of [3] and [13]. The only exception to this is NC and ND.

We obtain:

Theorem: A sentence of N is a theorem of the above axiomatisation if and only if it is valid.

Corollary: N is decidable.

The logic is, in Chellas' terminology, *normal* – that is, it is closed under the rules:

RCEA If $A \equiv B$ then infer $(A \Rightarrow C) \equiv (B \Rightarrow C)$.

RCK If $(B_1 \wedge \cdots \wedge B_n) \supset B$ then infer $((A \Rightarrow B_1) \wedge \cdots \wedge (A \Rightarrow B_n)) \supset (A \Rightarrow B)$ for every $n \geq 0$.

From this it follows that equivalent propositions can be substituted into the antecedent or consequent of a variable conditional, and from this an easy inductive argument shows we have full substitution of equivalents

From RCM and Id we obtain an unsurprising relation between implication and the variable conditional, namely:

RCE If $\vdash A \supset B$ then $\vdash A \Rightarrow B$.

We also obtain weakened forms for strengthening the antecedent and transitivity. In the first case we have, from NC:

CV $(A \Rightarrow B \wedge \neg(A \Rightarrow \neg C)) \supset (A \wedge C \Rightarrow B)$.

So if we have that ravens are (normally) black, but know nothing about albinoism, we can't conclude anything about albino ravens. However from CV we have that if it isn't the case that ravens are (normally) albino, then we can conclude that ravens that aren't albino are (normally) black.

We have already seen one valid form of transitivity:

If $\vdash A \Rightarrow B$ and $\vdash B \supset C$ then $\vdash A \Rightarrow C$.

This, from the proof-theoretic end of things, is a simple consequence of RCM. However we also have restricted transitivity in the form of RT.

$A \Rightarrow B \supset (A \wedge B \Rightarrow C \supset A \Rightarrow C)$.

Thus if ravens are (normally) black and ravens that are black (normally) have dark eye pigment, then ravens (normally) have dark eye pigment.

The logic N differs from conditional logics for counterfactual reasoning primarily in that these other systems generally have one or both of:

MP $(A \Rightarrow B) \supset (A \supset B)$

CS $(A \wedge B) \supset (A \Rightarrow B)$.

(See [8] and [13] for taxonomies of such systems.) If we were to add MP to N, we would obtain:

$\vdash A \supset B$ iff $\vdash A \Rightarrow B$

and so the variable conditional would collapse into entailment. CS requires that if the antecedent and consequent happen to be true at a world, then that world is a member of the least exceptional worlds with respect to the antecedent. For such worlds the variable conditional would have the same truth value as the material conditional Neither formula then is appropriate in the context of natural kinds. Conditional logics of obligation similarly differ in detail from N, most notably in rejecting the axiom Id.

6. A First-Order Logic for Natural Kinds

This work extends easily to allow quantification over variables. However it turns out that the obvious extension is not the most appropriate one. For example we have been representing "ravens are black" by

$$Raven \Rightarrow Black.$$

In first order terms the obvious representation is

$$(x)(Raven(x) \Rightarrow Black(x)).$$

But this isn't quite what we want. What we want to say is something like "in the normal course of events all ravens are black". What the above in fact says is that for each object x, in the normal course of events, if x is a raven then it is black. That is, we effectively have an arbitrary number of variable conditionals (one for each object) rather than a single attribution to a class.

A second alternative, developed in [5], is to introduce a monadic operator N where $N\alpha$ is read as "in the normal course of events, α" or "all other things being equal, α". The previous example would now be expressed as:

$$N(x)(Raven(x) \supset Black(x)).$$

This alternative follows from the observation that in the propositional case, the statement $A \Rightarrow B$ is equivalent to $M(A \wedge B \wedge L(A \supset B))$ in Sobocinski's K3. Similarly,

$$N(x)(A(x) \supset B(x))$$

can now be defined as

$$M(\exists x[A(x) \wedge B(x)] \wedge L(x)[A(x) \supset B(x)]).$$

The axioms of N can now be rewritten in first-order terms, so that, for example, CC becomes

$$(N(x)[A(x) \supset B(x)] \wedge N(x)[A(x) \supset C(x)]) \supset N(x)[A(x) \supset B(x) \wedge C(x)].$$

The axioms of N, so rewritten in first-order terms, are easily shown to be valid in K3, and so whatever we could say in the logic N, we can now state in first-order terms. Moreover we can in addition state conditions such as ravens (normally) have two wings. Perhaps:

$$N(x)[Raven(x) \supset \exists y,z(Left\_wing(x,y) \wedge Right\_wing(x,z))]$$

Lastly, it should be noted that the sentence $N\alpha$ where α isn't a universal material conditional, simply reduces to $ML\alpha$.

7. Discussion

This paper has presented a logical system N for dealing with statements concerning natural kind terms. It was developed by adding a "variable conditional" $\Rightarrow$ to standard propositional logic. The informal reading of $A \Rightarrow B$ is "all other things being equal, if something satisfies A then it satisfies B".

In this logic one can consistently assert, for example, that:

$Raven \Rightarrow Black$

$Raven \wedge Albino \Rightarrow \neg Black$

$Raven \wedge Albino \wedge Victim\_of\_Oil\_Spill \Rightarrow Black$

or that:

$Penguin \supset Bird$

$Penguin \Rightarrow \neg Fly$

$Bird \Rightarrow Fly$

However restricted versions of strengthening the antecedent and transitivity of the conditional are valid in N. These properties (arguably) conform to common intuitions concerning natural kinds.

Arguably also this approach is more appropriate for representing information about such kinds than are default logics or non-monotonic logics, in that its semantics does not rest on the notion of consistency with a given body of assertions. Thus the relation between ravens and blackness is phrased independently of any particular believer or believers. The system is decidable, and is presently being implemented using the method of semantic tableaux.

An additional area for further investigation is the relation of this approach to work in linguistic semantics. For example, Gregory Carlson in recent work [1], [2] has considered the problem of building a semantics for *generic* (as opposed to *episodic*) sentences such as "birds fly" and "the dodo is extinct". He proposes the framework of Montague grammar for treating generics. The approach at hand seems suitable for handling *indefinite singular* generic statements, which refer to properties that characterise members of kinds. Thus for example, the approach seems suitable for

Birds fly.

and, perhaps with some modification, for

Birds lay eggs.

It may be suitable for *habitual* sentences such as

John smokes.
John handles the mail arriving from Antarctica.

However it almost certainly isn't suitable for the *plural* and the *definite* generic, which can be used to refer to properties of a kind as a whole, for example,

Ravens are common.
The dodo is extinct.

Finally, in comparison to non-monotonic logics and systems of default reasoning, the work presented here contains one rather obvious omission: the system N is not in actual fact non-monotonic and does not contain any mechanism for reasoning "in the absence of other information". Thus for example if we know that A and that $A \Rightarrow B$, we have no mechanism for concluding something like "in the normal course of events, B". A similar problem is identified for conditional obligation in [20]. The resolution suggested there is applicable to the present case: that there is a separation between the principles of the calculus and the *application* of the calculus. Here we address the first case only.

This suggests a potentially useful merging of this work with that of default logics for dealing with information concerning natural kinds. If such information was to be represented, say, in a semantic network then a set of assertions concerning kinds could be *represented* in N: moreover the consistency of these assertions could be ascertained using this logic. However, an algorithm implementing a default logic such as is developed by Etherington and Reiter [6], could be used for *reasoning* about instances of these kinds. In the present case, the work of Etherington and Reiter would be directly applicable here since the default statements considered correspond in an obvious manner to statements of N.

Acknowledgments

I wish to thank Robert Hadley for his helpful comments on an earlier draft of this paper. I would also like to thank one of the referees for his/her detailed and pertinent comments

References

[1] G.N. Carlson, *Reference to Kinds in English*, Garland Publishing, 1980.

[2] G.N. Carlson "Generic Terms and Generic Sentences", *Journal of Philosophical Logic 11*, 1982, pp 145-181.

[3] B.F. Chellas, "Basic Conditional Logic", *Journal of Philosophical Logic 4*, 1975, pp 133-153.

[4] O. Dahl, "On Generics", in *Formal Semantics of Natural Language*, E.L. Keenan (ed.), Cambridge University Press, 1975, pp 99-111.

[5] J.P. Delgrande, "Logics for Natural Kinds", Technical Report 86-6, School of Computing Science, Simon Fraser University, 1986.

[6] D.W. Etherington and R. Reiter, "On Inheritance Hierarchies with Exceptions", *Proc. AAAI-83*, 1983, pp 104-108.

[7] G.E. Hughes and M.J. Cresswell, *An Introduction to Modal Logic*, Methuen and Col. Ltd., 1968.

[8] D. Lewis, *Counterfactuals*, Harvard University Press, 1973.

[9] J. McCarthy, "Circumscription -- A Form of Non-Monotonic Reasoning", *Artificial Intelligence 13*, pp 27-39, 1980.

[10] D. McDermott and J. Doyle, "Non-Monotonic Logic I", *Artificial Intelligence 13*, 1980, pp 41-72.

[11] R.C. Moore, "Semantical Considerations on Nonmonotonic Logic", *Proc. IJCAI-83*, Karlsruhe, 1983, pp 272-279.

[12] D. Nute, "Counterfactuals", *Notre Dame Journal of Formal Logic, Vol 16*, 1975, pp 476-482.

[13] D. Nute, *Topics in Conditional Logic*, Philosophical Studies Series in Philosophy, Volume 20, D. Reidel Pub. Co., 1980.

[14] H. Putnam, "Is Semantics Possible?" in *Mind, Language and Reality: Philosophical Papers Volume II*, Cambridge University Press, 1975, pp 215-271.

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

[16] E. Rosch, "Principles of Categorisation" in *Cognition and Categorisation*, E. Rosch and B.B. Lloyds eds., Lawrence Erlbaum Associates, 1978.

[17] B. Sobocinski, "Remarks About the Axiomatisations of Certain Modal Systems", *Notre Dame Journal of Formal Logic, Vol 5*, 1964, pp 71-80.

[18] R.F. Stalnaker "A Theory of Conditionals", in *Studies in Logical Theory*, N Rescher (ed.), Basil Blackwell, Oxford, 1968, pp 98-112.

[19] T.R. Thompson, "Parallel Formulation of Evidential-Reasoning Theories" *Proc. IJCAI-85*, Los Angeles, 1985, pp 321-327.

[20] B.C. van Fraassen, "The Logic of Conditional Obligation", *Journal of Philosophical Logic 1*, 1972, pp 417-438.

Fagin and Halpern on Logical Omniscience: A Critique with an Alternative

Robert F. Hadley

School of Computing Science, Simon Fraser University
Burnaby, Canada V5A 1S6

Abstract

Logical omniscience may be described (roughly) as the state of affairs in which an agent explicitly believes anything which is logically entailed by that agent's beliefs. It is widely agreed that humans are not logically omniscient, and that an adequate formal model of belief, coupled with a correct semantic theory, would not entail logical omniscience. Recently, two prominent models of belief have emerged which purport both to avoid logical omniscience and to provide an intuitively appealing semantics. The first of these models is due to Levesque [1984b]; the second to Fagin and Halpern [1985]. It is argued herein that each of these models faces serious difficulties. Detailed criticisms are presented for each model, and an alternative semantic theory is outlined which, it is argued, exposes the causes of the problem of logical omniscience, and provides its solution.

Introduction

A noticeable trend among AI researchers who seek to formalize the concepts of knowledge and belief has been to adopt denotation-based (or truth-based) semantic theories, which follow the spirit, if not the letter, of Tarski's method of interpreting formal languages. The danger[1] of construing Tarski's method as a semantic theory, however, has fostered an interest in more sophisticated semantic theories, such as "possible world" semantics. Whereas denotational semantics attempts to fix the meaning of a set of sentences by reference to their denotations in the *actual* world, "possible world" semantics attempts to fix such meaning in terms of denotations (including truth values) in all logically *possible* worlds.

It is known, however, that *some* versions of possible world semantics have strongly counter-intuitive consequences, one of which has come to be known as the problem of *logical omniscience* [Hintikka, 1975]. Briefly stated, to assert that X is logically omniscient is to assert that X believes all the logical consequences of any of X's beliefs, and furthermore that X believes all logically valid sentences, regardless of the complexity of those sentences.

Now, there is general agreement that any theory which entails logical omniscience is inadequate as an account of belief. It is not clear, however, that all variants of possible world semantics are committed to logical omniscience. For, strictly speaking, a semantic theory only provides an account of how meanings are assigned to symbolic expressions. A particular theory may provide an analysis of what meanings are, and how expressions come to be synonymous, and yet say nothing about what sentences are believed, or when expressions are interchangeable in belief contexts. For example, one may assert that sentences are synonymous iff they are true in the same set of possible worlds, and still deny that if sentence S is believed, then any sentence synonymous with S must also be believed. Of course, the awkward conclusion that all tautologies are synonymous would still be entailed by the possible world semantics just envisioned.

The doctrine of logical omniscience becomes more difficult to separate from a "possible world" semantics when that semantics is wedded to an axiomatic model of belief, as in [Hintikka, 1962]. In what follows we examine two theories, which combine differing approaches to possible world semantics with axiomatic belief models, and which purport to avoid logical omniscience. The first of these theories, due to Levesque [1984b], is embedded in a general account of explicit and implicit belief, in which the focus shifts from "possible worlds" to partial possible worlds (or situations). In section 1 we discuss Levesque's approach in some detail, and argue that it is committed to a class of counter-intuitive results, e.g., that anyone who explicitly believes that there are 97 apples on the table must explicitly believe that the number of apples on the table is the largest prime number less than 100.

In section 2 we examine an axiomatic model of belief and awareness which is due to Fagin and Halpern [1985]. This theory integrates a possible world approach with a "syntactic awareness" function. On this theory one cannot believe a sentence to be true unless one is "aware" of that

[1] Tarski's method was not intended as a theory of meaning, but it has sometimes been regarded as such. That it should not be so regarded becomes apparent when we reflect that "identity of denotation" does not entail "identity of meaning". Otherwise, 'unicorn' would be synonymous with 'round square', and 'featherless biped' would be synonymous with 'human being'.

sentence. It will be argued that Fagin and Halpern's approach is committed to conclusions which are nearly as difficult to accept as the doctrine of logical omniscience itself. Among these are:

- That one already believes any formula which happens to be tautologous, even though one has just become aware of and understood the formula, but not yet discovered it to be true.
- On a natural and common interpretation of "awareness", one cannot believe both p and $p \rightarrow q$ without believing q.

In sections 3 and 4 an alternative semantic theory is described which, it is argued, solves the problem of logical omniscience, and still accords well with our intuitions. This semantic theory is implicit in approaches taken by some workers in natural language understanding (e.g., [Winograd, 1973; Dahl, 1981; Hadley, 1985]). The particular theory proposed here is a hybrid of principles presented in [Lewis, 1976] and [Hadley, 1973]. Unlike the theories discussed in sections 1 and 2, this theory emphasizes the *procedural* aspect of meaning, and provides an explanation of how two sentences may have the same meaning, and yet not be interchangeable in belief contexts. It also explains why tautologies, which are coextensive in all possible worlds, need not be synonmous.

1. Levesque's Model

We turn now to consider Levesque's theory of explicit and implicit beleif, and to consider how it seeks to solve the problem of logical omniscience [Levesque, 1984b]. The theory incorporates a modified situational semantics [Barwise and Perry, 1983]. The critical difference between Levesque's approach and a typical possible world approach is that Levesque restricts the objects which support beliefs to *situations* (or partial worlds) rather than to complete possible worlds. We may think of a situation as a fragment of a possible world which is relevant to the truth of certain primitive propositions in our object language, but not to others. For example, the situation we intuitively describe as eating one's breakfast has no bearing upon the truth of the proposition "There are 200 words in the Canadian national anthem." The fact that some situations have no bearing on the truth *or* falsity of certain primitive propositions is crucial to the way in which Levesque deals with logical omniscience.

Levesque creates a formal model for explicit and implicit belief by introducing a language, L, which contains atomic propositions, standard truth-functional connectives, and the unary connectives B and L. The connective B may be prefixed to any well-formed sentence α, of L, provided α contains no occurrences of B or L. Analogous remarks apply to L. Intuitively, we may interpret Bα as "α is believed" and Lα as "α is logically implied by something which is believed". Levesque also equates Lα with "α is implicitly believed", which is, perhaps, a contentious use of "implicit belief".

The semantics of the operators B and L and of the "support" relations $\models_t$, and $\models_f$, are partially constrained by axiom schemata. These axiom schemata are semantically interpreted in terms of a model structure **<S, B, T, F>**, where the intended interpretations of **S, B, T,** and **F** are as follows:

- **S** is to be taken as the set of all possible situations,
- **B** is the set of all situations which *could be the actual situation* according to what is believed,
- **T** and **F** are functions such that: **T** maps any primitive proposition p, of L, into those situations from **S** which support the truth of p, and **F** maps p into those situations which support the falsity of p.

Shown below is a subset of Levesque's axioms which are relevant to our discussion. The support relations, $\models_t$ and $\models_f$, may be interpreted as follows: if s is a situation and α is a sentence of L, then $s \models_t \alpha$ iff s supports the truth of α, and $s \models_f \alpha$ iff s supports the falsity of α. (For brevity, we shall render 'supports the truth of' as 'confirms', and 'supports the falsity of' as 'disconfirms'. No claim is made that these expressions are synonymous in natural language. Rather, we shall merely use "confirms" and "disconfirms" as technical abbreviations for the longer expressions.)

1. $s \models_t p$ iff $s \in \mathbf{T}(p)$.
   $s \models_f p$ iff $s \in \mathbf{F}(p)$.

2. $s \models_t (\alpha \vee \beta)$ iff $s \models_t \alpha$ or $s \models_t \beta$.
   $s \models_f (\alpha \vee \beta)$ iff $s \models_f \alpha$ and $s \models_f \beta$.

3. $s \models_t (\alpha \wedge \beta)$ iff $s \models_t \alpha$ and $s \models_t \beta$.
   $s \models_f (\alpha \wedge \beta)$ iff $s \models_f \alpha$ or $s \models_f \beta$.

4. $s \models_t \neg\alpha$ iff $s \models_f \alpha$.
   $s \models_f \neg\alpha$ iff $s \models_t \alpha$.

5. $s \models_t B\alpha$ iff for every s' in **B**, $s' \models_t \alpha$.
   $s \models_f B\alpha$ iff $s \not\models_t B\alpha$.

Fagin and Halpern [1985] discuss certain technical and philosophical difficulties arising from the model presented above. We shall not discuss those difficulties here, since Levesque is reported to be dealing with the most serious of them in [Levesque, in progress]. Rather, we consider a separate problem which appears to be intrinsic to any approach, like Levesque's, which attempts to individuate beliefs on the basis of the situations which "support" those beliefs. The problem arises when two sentences must both be treated as primitive with respect to the language L,

and both sentences are supported by identical situations. Consider the following pair of sentences:

(a) The number of apples on the table equals the largest prime number less than 100.

(b) The number of apples on the table equals 97.

In the language L, both sentences must be treated as primitive, since neither can be reduced to a *truth-functional* compound of other primitive sentences within L. Yet, a moment's calculation reveals that 97 *is* the largest prime less than 100, and consequently, that both these sentences describe the same situation (or state of affairs). Therefore, any reasonable semantic functions, **T** and **F**, would assign (a) and (b) to identical situations, though both are primitive sentences. Yet, it is clear that a person or system could believe (b) without believing (a). It is even possible to believe (b) without having the concept of a prime number. But, on Levesque's model, whoever believes (b) must believe (a). This is apparent from the fact that, according to axiom 5, (b) could not be believed unless every situation s' in **B** confirms (b). And this, in turn, could only be true if every situation s' in **B** confirms (a), since **T**(b) = **T**(a), and axiom 1 holds.

We seem forced to conclude that beliefs cannot be individuated by the identity of situations which confirm those beliefs. The following objection may arise, however. In the foregoing discussion (a) and (b) were both treated as primitives. But, in a language richer than L, (a) might well be treated as a composite sentence, while (b) is treated as a primitive. For this reason, it is unfair to represent both (a) and (b) as primitives in L. Perhaps the most reasonable course is to exclude sentences such as (a) from the scope of language L, on the grounds that they express complex propositions which are not representable in a propositional calculus.

To reply to this objection, we should first reflect that if sentences such as (a) are not permitted to be primitive sentences in L, then the power and range of L is less than we may have imagined. Second, we will eventually need a formal model, M, in which both (a) and (b) are expressible. In the model M, (a) and (b) may not both be primitive, but if M is to be given an intuitively plausible interpretation, both sentences will *still* be confirmed by identical situations, because they obviously *describe* identical situations. In this case, it would still be impossible to distinguish the belief that (a) from the belief that (b) solely on the basis of the situations which confirm those assertions.

2. The Fagin-Halpern Model

We have remarked that in "Belief, Awareness, and Limited Reasoning" [1985], Fagin and Halpern voice criticisms of Levesque's theory which differ from those described here. However, they proceed to offer two different models of belief and awareness which are, to varying degrees, modifications of Levesque's approach. Both models are generalized to permit sentences to contain nested belief operators, and to accomodate agents who have separate beliefs and beliefs about each others beliefs. These generalizations are important and interesting, but shall not concern us here. Rather, we are concerned with the manner in which Fagin and Halpern handle the problem of logical omniscience. The first of their models is clearly in the spirit of Levesque's approach, but suffers (as the authors concede) from the defect that it requires an agent's beliefs to be closed under implication. The second model avoids this defect (in the authors' view) while retaining the improvements found in the first model. For this reason, we will confine our attention to the second model, M, which Fagin and Halpern refer to as a Kripke structure for general awareness.

In M, Levesque's belief operators, B and L, are relativized to individual agents, so that we may write $B_i\alpha$, meaning that agent$_i$ explicitly believes α, and $L_i\alpha$, meaning that α is implied by what agent$_i$ believes. Following Levesque, L_i is called an implicit belief operator. Now, one respect in which Fagin and Halpern differ from Levesque is that they make the notion of truth - relative to a complete state of a possible world - central to the definition of explicit belief. On their theory, **S** is to be taken as the set of all possible world states, and $\mathbf{B}_i$ is a binary relation on **S** which is transitive, Euclidean, and serial. Intuitively, $\mathbf{B}_i$ is such that, for any states s and t (members of **S**), the pair (s,t) is in $\mathbf{B}_i$ if t is a possible world state according to what agent$_i$ believes in state s. Note that, in model M, agents are not permitted to have incoherent beliefs. Thus, each s in **S** is a coherent state.

In the model M there is a truth assignment function π, which assigns truth values to each primitive proposition p, relative to state s. Thus, for each pair s and p, $\pi(s,p) \in \{true, false\}$. The logic also includes an awareness operator A_i which is applied to formulas. For example, $A_i\alpha$ could be read "agent$_i$ is aware of formula α". Fagin and Halpern emphasize that they do not wish to place restrictions upon what counts as awareness, but they clearly indicate that awareness of a formula does not, in general, entail belief in its truth. For example, one may be aware of Fermat's last theorem without believing it.

For each agent$_i$ a set of formulas $\mathbf{A}_i(s)$ is specified, and is to be taken as the set of all formulas which agent$_i$ is aware of in state s. Also, the logic includes a two-valued truth relation, $\models$, which is implicitly defined by the following axioms, completing the model:

1. $M,s \models p$, where p is a primitive proposition, if $\pi(s,p)$ = true.

2. $M,s \models \neg\phi$ if $M,s \not\models \phi$.

3. $M,s \models \phi_1 \wedge \phi_2$ if $M,s \models \phi_1$ and $M,s \models \phi_2$.

4. $M,s \models A_i\phi$ if $\phi \in \mathbf{A}_i(s)$.

5. $M,s \models B_i\phi$ if $\phi \in \mathbf{A}_i(s)$ and $M,t \models \phi$ for all t such that $(s,t) \in \mathbf{B}_i$.

6. $M,s \models L_i\phi$ if $M,t \models \phi$ for all t such that $(s,t) \in \mathbf{B}_i$.

(The capital 'M', which occurs in each axiom, primarily serves to remind us that the relation $\models$ pertains only to the model M.)

Given these axioms, it can be seen that, for any tautology, ψ, and any state s, $s \models \psi$. This becomes apparent once we reflect that *if*, for some s, $s \not\models \psi$, then by axiom 2 we would have that $s \models \neg\psi$. But, since $\neg\psi$ must be self-contradictory, we would have that state s verifies the truth of an inconsistent sentence, which is clearly impossible, given axioms 1-3, and that each s represents a *possible* world state. In light of axiom 6, and the fact that every state satisfies the $\models$ relation to any tautology, we may infer that $L_i\psi$, for every tautology ψ. This explains what lays behind Fagin and Halpern's remark that "L_i acts like the classical belief operator."

Given the above, it follows that an agent believes any tautology that the agent is merely "aware of". For if $agent_i$ is aware of a tautology ψ, then $\psi \in \mathbf{A}_i(s)$, and since $t \models \psi$ for any state t, we know that $t \models \psi$ for all t such that $(s,t) \in \mathbf{B}_i$. Thus, by axiom 5 we have that $B_i\psi$. Fagin and Halpern clearly notice this ramification, for they remark that "Agents still do not explicitly believe all valid formulas; for example, $\neg B_i(p \vee \neg p)$ is satisfiable because the agent might not be aware of the formula $p \vee \neg p$." The clear implication is that $\neg B_i(\phi)$ is only satisfiable for valid ϕ if the agent is not aware of ϕ.

The question now arises, is this implication of model M substantially more plausible than the doctrine of logical omniscience? We shall argue that it is not. Although Fagin and Halpern refrain from defining "awareness", they do say that it is not to be equated with belief in general, and they suggest a number of possible interpretations for $A_i\phi$. Their first interpretation amounts to "mere awareness", with no implication that belief might follow. A second interpretation is "i is able to figure out the truth of ϕ". A third suggestion is "i is able to compute the truth of ϕ within time T." Now on any one of these interpretations it is clearly not true that one actually believes every tautology one is aware of. Consider an adept logic student who is given a list of wffs and is told to determine which are tautologous. The student picks a complex one which happens to be tautologous, and begins constructing a truth table for it. At this point the student may well be "aware of" the wff in all three of the above senses, but has not yet determined whether the wff is valid. Even after completing the truth table the student may not believe the wff to be valid until all work has been rechecked. It is clear that awareness is only one ingredient in coming to *believe* a valid wff; both adequate computational resources and confidence in one's abilities are also required. Perhaps confidence may be ignored when dealing with automata, but the issue of computational resources cannot.

Fagin and Halpern discuss the importance of computational resources in connection with Levesque's semantics, when they say "There may well be a very complicated formula whose truth is hard to figure out, even if you are aware of all the *primitive propositions* that appear in it" (my italics). Apparently, the authors have not realized that this remark remains true when we replace the "even if..." clause with "even if you are aware of the entire formula."

Apart from the above difficulty, a different problem arises for the Fagin-Halpern model, namely, on a natural and common interpretation of "awareness", an agent's *explicit* beliefs are *closed* under implication. To understand why this is so, we first note that within the logic of M, $B_ip \wedge B_i(p \rightarrow q)$ implies B_iq, unless i is unaware of the formula q.[2] Fagin and Halpern acknowledge this, but maintain that "$B_ip \wedge B_i(p \rightarrow q) \wedge \neg B_iq$ is satisfiable since i might not be aware of q." But we argue that, on the usual interpretation of awareness, one cannot believe $(p \rightarrow q)$ without being aware of q, and hence, that belief is closed under implication.

To begin with, one could not believe the formula '$p \rightarrow q$' to be true unless one *understood* the formula, and one cannot understand the formula without understanding its parts. (Imagine our response to someone who claimed to understand '$p \rightarrow q$' but who did not understand the formula 'q'.) But if one understands 'q', then one must at least be aware of 'q' as a formula. Consequently, belief that '$p \rightarrow q$' is true entails awareness of the formula 'q', on a natural interpretation of awareness. This is not to deny that on *some* unusual interpretation of "awareness" one might believe '$p \rightarrow q$' and be unaware of the formula q. For example, if we interpret "i is aware of ϕ" as "i can compute the truth of ϕ in time T," (an interpretation suggested by the authors) then it may be possible to believe '$p \rightarrow q$' is true without being aware of q. However, while there is no formal requirement within the model that "awareness" be given a natural interpretation, it is important to realize that there exists a natural and obvious interpretation of "awareness" under which explicit belief remains *closed* under logical consequence.

Furthermore, unless "awareness of ϕ" is assigned a natural and intuitive interpretation, it becomes difficult to accept axiom 5, which states, among other things, that

[2] It can only happen that $B_ip \wedge B_i(p \rightarrow q)$ if, for all t such that $(s,t) \in \mathbf{B}_i$, $(t \models p) \wedge (t \models (p \rightarrow q))$. This, in turn, requires that for all such t, $t \models q$. (Otherwise t is an inconsistent world state). But now, if $agent_i$ is aware of q we have that $q \in A_i(s)$, and we could derive B_iq. Therefore, $\neg B_iq$ requires that i not be aware of q.

"belief in ϕ" entails "awareness of ϕ". After all, one may believe Fermat's last theorem without being able to compute its truth in any given time span, so why should we accept axiom 5 when "awareness' is equated with computability, or some other unusual interpretation?

A final difficulty with the Fagin-Halpern model is that, since it links belief to awareness of particular formulas, it shares those aspects of the "syntactic approach" which are criticised by Levesque [1984b]. While Fagin and Halpern attempt to answer these criticisms, there is a problem with the syntactic approach which they do not consider. This problem arises in cases such as the following:

(a) All mules are stubborn.

(b) All mules are obstinate.

Suppose that X knows the word 'stubborn' but not the word 'obstinate'. Many philosophers would still contend that if X believes what (a) asserts, then X believes what (b) asserts, since both sentences express exactly the same proposition. The object of belief, they maintain, is a proposition, not a string of symbols (cf. [Church, 1954]). Fagin and Halpern defend the syntactic approach by claiming that formulas which are ostensively synonymous may involve different computational processes (viz., "$p \vee q$" vs. "$q \vee p$"), but it is difficult to see how this response would apply to the present case, since 'stubborn' and 'obstinate' seem to involve the same computational processes.

Furthermore, we shall argue in section 4 that $(p \vee q)$ and $(q \vee p)$ need *not* involve different computational processes. It is revealing, however, that Fagin and Halpern imply that computational processes are *relevant* to the question whether two sentences can express the same belief. This aspect of belief is central to the semantic theory which is outlined in the following two sections.

3. A Procedurally Based Semantics

The theory described below builds upon themes prevalent in the philosophy of language since the time of Frege. Aspects of the theory can be extracted from the works of Ayer [1936], Carnap [1947], and Church [1954]. Two central principles emerge from these works; they are:

(a) the meaning of a linguistic expression may be identified with *rules* which govern its use. Members of the logical-positivist school, e.g., Carnap and Ayer, construed these rules as the method of verifying the truth or falsity of an expression.

(b) the meaning of any declarative sentence, even atomic sentences, is a strict function of the meaning of its parts.

While both these principles have a certain intuitive appeal, philosophers have encountered considerable difficulty in defending the application of these principles to natural language. However, the philosopher, Richard Montague, did much to rectify this situation [Montague, 1970]. An elegant variation on Montague's approach is presented by Lewis in [Lewis, 1976]. Roughly, on both Montague's and Lewis's accounts, the intension of a sentence is a semantic function which takes a possible world state as an argument, and returns a truth value. This semantic function is formed through the *composition* of simpler functions which are attached to the more elementary elements of the language. The particular manner in which this composition happens is determined by the syntax of each sentence. From a *procedural* standpoint, the resulting composite function is isomorphic to the *deep structure* of the given sentence.

Now, on the face of it, a problem arises for this approach in the case of logically valid sentences. For example, since all tautologies are true, it may appear that the intension of each tautology is any function which returns true, given any possible world state. From a denotational standpoint, all such functions are identical, because they all have identical input-output relations. Consequently, if meaning is identified with denotationally construed functions, all tautologies become synonymous; which is strongly counter-intuitive.

Lewis avoids this defect by identifying meaning with a special syntax tree, where all nodes are labelled with functions (intensions), as well as with syntactic labels. This solution has a drawback, however, since it seems to invite Levesque's criticisms of the "syntactic approach." For example, '$p \vee q$' has a different syntax tree from '$q \vee p$', and consequently these would not be synonymous on Lewis's theory. Furthermore, as Levesque points out, what is the semantic motivation for linking meaning to the syntax of a sentence?

We have noted that Fagin and Halpern reject Levesque's criticism of the syntactic approach because they maintain, in effect, that '$p \vee q$' is not really synonymous with '$q \vee p$'. For some however, this reply remains unconvincing, since these expressions certainly *seem* synonymous. We now attempt to show that a small modification to Lewis's theory can explain *how* these expressions could be synonymous, and also explain the semantic motivation for the view that meaning is isomorphic to syntax.

The solution requires that we focus upon the procedural nature of the semantic functions involved. Functions which have identical input-output relations ("weak equivalence") need not be identical from the computational standpoint (e.g. compare heapsort to bubblesort). But, if we individuate functions by the criterion of "strong equivalence" [Pylyshyn, 1980], then in essence we distinguish functions according to the computational procedures they embody. From this perspective, a complex procedure will have its structure imposed through the composition of functions. Likewise, any computational procedure which *is* the intension of a sentence will be determined by compositional processes involving simpler procedures and by the syntax of the sentence involved.

For example, consider the sentence "John sleeps". The intension of 'John' is a function which takes a possible world (or an index of a possible world) as an argument and which returns a particular individual as its value. The intension of the word 'sleeps' may be a function which requires, as argument, a function of the above type, and which returns, as value, a more complex procedure which is the intension of the entire sentence. This composite procedure will require a possible world as argument, and return a truth value according to whether "John sleeps" is true in that world. The procedure has a structure of the form "f(g(x))", which reflects the manner in which it is composed. The fact that 'sleeps' has an intension which requires another intension as argument, is determined by the syntactic role of that verb.

We may now explain the semantic motivation for linking meaning to syntax. Given the Montague-Lewis approach, it is only reasonable to suppose that sentential intensions, *construed as procedures*, are isomorphic to syntax, since syntax enters into the process by which such procedures are composed. These intensions may be identified with "propositions".

The question now arises, how can this conclusion be reconciled with the fact that for some people at least, '$p \vee q$' is synonymous with '$q \vee p$'? To answer this, we must recall that procedures are, in general, more abstract than particular implementations of those procedures in a programming language. At the same time, there is an isomorphism between the high-level structure of such implementations and the abstract procedure they embody. Likewise, there is an isomorphism between all possible implementations of a given high-level programming language. Furthermore, just as a given high-level language may receive differing implementations without ceasing to be distinct from other high-level languages, so a particular algorithm, say heapsort, does not cease to be distinct from equivalent algorithms just because some of its implementation details are left unspecified.

Now suppose that the *meaning* of '$p \vee q$' is an abstract procedure which can be implemented in more than one way. For example, '$\vee$' may name a procedure whose sole argument is an abstract *set* of propositions. The "$\vee$" procedure attempts to verify that at least one element of this set is true. One implementation of this procedure attempts to establish the truth of p before attempting to establish q. Another implementation does the reverse. But if the meaning of '$\vee$' is the abstract procedure, and not one of its implementations, then both '$p \vee q$' and '$q \vee p$' will determine, through the compositional process, the same abstract procedure as their intension. Furthermore, since each sentence has a syntax isomorphic to the other, we might still maintain that the intension determined by a sentence is isomorphic to its syntax.

As a general requirement, however, strict isomorphism is a bit too stringent. For many would maintain that a simple declarative sentence in active voice is synonymous with its passive equivalent. Yet, it is not clear that "John threw a ball" is isomorphic in syntax to "A ball was thrown by John". To accomodate syntactic variants of a given sentence we may follow Lewis [1976] in supposing that the *deep structure* (and not the surface structure) of a sentence determines the process of intensional composition (since many linguists theorize that a sentence is first translated into its deep structure before its meaning is "understood".)

Another respect in which the "strict isomorphism" requirement must be relaxed is this: we must allow that two sentences express the same intension when both sentences are otherwise identical, but one contains a single term which is a definitional abbreviation of a longer expression, occurring in a corresponding position in the other sentence. For example, this relaxation will allow us to say that "John is a bachelor" is synonymous with "John is an unmarried adult male", since 'bachelor' is a conventional abbreviation for 'unmarried adult male'. Perhaps the accuracy of our particular definition of 'bachelor' could be disputed, but the general point remains that if a given term has come, by linguistic convention, to have the same sense as a longer expression, then each may be substituted for the other in a longer expression, without altering the intension of the longer expression. An analogous remark obviously applies to programming languages which allow "macros" or subroutines to be defined. The substitution of the name of a routine, for the routine itself, does not affect any larger procedure in which either occurs.

A similar modification is required to accomodate one-word synonyms (e.g., 'stubborn' and 'bachelor') which are attached, by convention, to the same intension. Clearly, such synonyms may be interchanged without affecting the intension of a larger, containing expression. Given all the above modifications we are now in a position to demonstrate how our procedure-based semantics avoids one major aspect of logical omniscience. We require the following premise:

(1) Two expressions are interchangeable in belief contexts *only if* they express identical intensions, procedurally construed.[3]

This premise has an intuitive appeal, since we have no reason to suppose that expressions which do *not* mean the same would be interchangeable in belief contexts. Now, given (1), it straightforwardly follows that logically valid sentences are not, in general, interchangeable in belief contexts. Consider the formulas:

$$(p \wedge q) \vee (\neg p \wedge q) \vee (p \wedge \neg q) \vee (\neg p \wedge \neg q)$$

and

$$(p \vee \neg p).$$

[3]Note that (1) states only a necessary condition for interchangeability. In section 4 we formulate a sufficient condition, in conjunction with our discussion of the paradox of analysis.

Both sentences are tautologous, and therefore equivalent. However, they are not isomorphic in syntax. Neither do they contain subexpressions such that one subexpression is a definitional abbreviation of the other. Furthermore, there is no reason to suppose that one sentence represents the deep syntactic structure of the other. Consequently, on the account of meaning we have presented, we have no reason to assume these formulas express identical intensions, and by (1), no reason to assume interchangeability in belief contexts. We may extend this reasoning to any pair of valid sentences which meets the above conditions.

4. A Solution to Logical Omniscience

In the preceding section we have given necessary conditions for allowing expressions to be interchanged in belief contexts. In what follows we formulate sufficient conditions. Finally, we argue that this account of necessary and sufficient conditions entirely avoids the problem of logical omniscience.

At first glance it might appear that sameness of intension should be a sufficient condition for allowing any two expressions to be interchanged in belief contexts. However, this formulation is too course-grained, since there are synonymous expressions which are *not* interchangeable in belief contexts. Consider a case where Jane believes that Eve is an ancestor of John. An analysis of the meaning of 'ancestor' may reveal that (a) "X is an ancestor of Y" is definitionally equivalent to (b) "either X is a parent of Y, or X is a parent of an ancestor of Y." The analysis may be based upon empirical evidence that when people discover, first hand, whether X is an ancestor of Y, they in fact use this recursive procedure to decide the question.[4] Let us suppose this is so. Then, on our account, the given expressions are synonymous. Yet, it may happen that some English speakers are *not fully aware of the procedure* that they apply when deciding whether X is an ancestor, and so they may not realize the synonomy of (a) and (b). Such speakers may believe (a) and not believe (b), and considerable reflection may be required to discover the synonomy involved. The critical point here is that one's use of an expression may be implicitly governed by a procedure without one's explicitly knowing what the procedure is.[5] This fact gives rise to what G.E. Moore has called the "paradox of analysis" [Moore,1942].

We have just seen that synonomy is too weak a condition to legitimize interchangeability in belief contexts. Also, in light of Levesque's criticisms of the "syntactic approach", we may rule out strict identity of expressions as too stringent a condition for interchangeability. We would like a set of conditions which permit the interchangeability of

$$(p \vee q) \quad \text{with} \quad (q \vee p)$$

and

'all mules are stubborn' with
'all mules are obstinate'

while ruling out the interchangeability of (a) with (b). To this end we propose the following criteria:

(C) **Two expressions are interchangeable in belief contexts iff they have identical intensions, procedurally construed, *and* they are isomorphic in syntax.**

Given these criteria, we may establish that

(i) from the fact that an agent believes *some* valid formula we may not derive that the agent believes all valid formulas.

(ii) one does not necessarily believe $p \vee \neg p$.

(iii) belief is not closed under implication.

The truth of (i) was established in section 4, using the criterion of sameness of procedural intension. The truth of (ii) follows from the fact that our theory makes no claims about what propositions are believed. It only specifies conditions under which expressions are interchangeable in belief contexts. Our view is that, insofar as a logic of belief exists, it must be restricted to domains where identical beliefs are being substituted for one another. In general, the question whether one believes p, given that one believes q, and that p and q do not express identical beliefs, seems not to be a question of logic but of empirical psychology. A exception to this claim may be that from the fact that a conjunction is believed we may infer that each conjunct is separately believed.

With regard to closure under implication (iii), it is apparent that no matter what procedures (intensions) we assign to p and $p \rightarrow q$, the intension of q may be distinct from either of the former intensions. Given that an agent stands in the "belief relation" to the intension of p and to the intension of $p \rightarrow q$, there is no way to deduce, given our criteria, that one must stand in this relation to the intension of q. Thus we avoid logical omniscience.

Conclusion

In the foregoing we have examined two differing formal models of belief which are designed to circumvent logical omniscience. The first model (Levesque's) attempts to individuate beliefs on the basis of the situations they

---

[4] For a discussion of how such empirical evidence may be gathered, see [Pylyshyn, 1980].

[5] We have seen that Levesque defines "implicit belief" in such a way that *every* logical consequence of a set of beliefs is implicitly believed. The above account provides what we believe to be a more plausible use of "implicit knowledge or belief".

describe. We have argued, however, that situations are too coarse-grained to serve this purpose, since different descriptions may *necessarily* describe the same situation, and still not be interchangeable in belief contexts.

The second model (Fagin's and Halpern's) combines a possible worlds approach with an awareness requirement. We have seen that this synthesis entails a restricted version of logical omniscience, insofar as it requires that an agent believes any valid formula which the agent is merely aware of. In addition, we have argued that it shares a weakness with the "syntactic approach".

Finally, we have presented an integration of some existing philosophical approaches towards semantics. We have argued that this integration provides a foundation for a sound principle of individuation for beliefs, while also providing a computational model of propositions. A formal logic of belief has not been presented, but it is unclear whether a logic of explicit belief, which sanctions the derivation of new beliefs from other, distinct beliefs, is even possible.

References

Ayer, A.J., (1936) *Language, Truth and Logic*, Gollancz Press.

Barwise, J. and Perry, J., (1983) *Situations and Attitudes*, Bradford Books, Cambridge.

Carnap, R., (1947) *Meaning and Necessity: A Study in Semantics and Modal Logic*, University of Chicago Press.

Church., A., (1954) "Intensional Isomorphism and Identity of Belief", *Philosophical Studies*, 1954.

Dahl, V., (1981) "Translating Spanish into Logic Through Logic", *American Journal of Computational Linguistics*, Vol. 7, No. 3.

Fagin, R., and Halpern, J., (1985) "Belief, Awareness, and Limited Reasoning", Proceedings of IJCAII, Los Angelos, August, 1985.

Frege, G. (1952) "On Sense and Reference", in *Translations from the Philosophical Writings of Gottlob Frege*, Oxford.

Hadley, R.F., (1973) *Convention and the Intensional Concepts*, Ph.D. Thesis, Dept. of Philosophy, University of British Columbia.

Hadley, R.F., (1978) "Possibility, Necessity, and Logical Truth", *Analysis*, Vol. 38, No. 4.

Hadley, R.F., (1985) "SHADOW: A Natural Language Query Analyser", *Computers and Mathematics with Applications*, Vol. 11, No. 5.

Hintikka, J., (1962) *Knowledge and Belief: An Introduction to the Logic of the Two Notions*, Cornell University Press.

Hintikka, J., (1975) "Impossible Possible Worlds Vindicated", *Journal of Philosophical Logic*, 4, 1975.

Levesque, H.J., (1984a) "A Logic of Implicit and Explicit Belief", Proceedings of AAAI-84, Austin, 1984.

Levesque, H.J., (1984b) "A Logic of Implicit and Explicit Beleif", a revised version of the above, FLAIR Tech. Report #32.

Levesque, H.J. (in preparation) "Global and Local Consistency and Completeness of Beliefs".

Lewis, D. (1976) "General Semantics", in *Montague Grammar*, (ed.) Partee, B., Academic Press, New York.

Montague, R., (1970) "Universal Grammar", *Theoria* 36.

Moore, G.E., (1942) "Reply to My Critics", in *The Philosophy of G.E. Moore*, (ed.) Schilpp, P., Evanston.

Pylyshyn, Z., (1980) "Computation and Cognition: Issues in the Foundations of Cognitive Science", *Behavioural and Brain Sciences*, Vol. 3, No. 1.

Winograd, T., (1973) "A Procedural Model of Language Understanding", *Computer Models of Thought and Language*, (eds.) Schank and Colby.

Representing Contextual Dependencies in Discourse

Tomek Strzalkowski

School of Computing Science
Simon Fraser University
Burnaby, B.C. V5A 1S6, CANADA

Abstract

We present a fragment of a formal theory for computing and representing a class of contextual dependencies in natural language discourse. Four types of contextual situations created by the use of definite descriptions are investigated, and formal translation rules into a lambda-calculus based meaning representation language Λ are introduced

1. Introduction

In this paper we present a fragment of a formal theory for computing and representing a class of contextual dependencies in natural language discourse Although we limit our discussion to anaphoric references created by the use of definite descriptions, the approach can be easily extended to cover other types of anaphora (such as those listed in [13]), as well as more difficult reference cases including forward references and indirect references. We investigate here only four types of contextual situations which we call *perfect contexts*, *imperfect contexts*, *attitude report contexts*, and *conditional contexts*. For each of these types we formulate a translation rule into the lambda-categorial language Λ, see for example [3]. The limited scope of this presentation has not allowed for including the description of Λ's syntax neither for formal definition of the translation procedure. These omisions should not obscure the readability of the paper, however, which is otherwise self-contained. This paper is based on a fragment of a more general Theory of Stratified Meaning Representation [10] [12] where a wider range of problems of natural language understanding is addressed including interpretation and representation of pronominal references, non-singular concepts, as well as selected problems of discourse coherence and building discourse representation.

An important aspect of our method is its computational perspective. Although the translation scheme we present is based in part on Montague's approach [6], we made an effort to replace intensional interpretations of certain types of expressions, such as *John is looking for a unicorn*, or *John ordered a hamburger* [5], by a more careful analysis which allowed for more tractable representation. This does not mean that we discard intensional interpretations altogether, but we feel that the concept has been much overused, and the sentences like the two quoted above are not truly intensional, i.e., we do not need the intension operator to represent their meaning. The use of the intension operator would be appropriate for representing such non-singular concepts like *the temperature* in *the temperature is increasing*.

In our discussion we focus on selected examples of two-sentence "stories" and try to discover and formalize referential interdependencies between them and the conditions in which such dependencies arise. Restricting the discussion to two sentence "stories" avoids, at this stage, most of the problems of *where to look for the reference*, i.e. how to determine the proper antecendent thus we concentrate entirely on the question *how to get the reference*, that is, how to select possible antecendents The fact that we restrict our discussion to two-sentential paragraphs should not be taken literally, especially when interpreting the translation rules we formulate in this paper. Most rules will be written using two hypotetical "sentences" S_1 and S_2 where the latter contains a reference element to an object mentioned in the former The sentence S_1 will normally be considered to establish a context for making that reference. In fact it is not necessary that any single such sentence exists. We assume only that at the time S_2 is expressed there is enough context information accumulated by different means so that the reference can properly be made. In our idealized model all this is reduced to the situations described by the sentence S_1 or its elements, but note that, in general, S_1 need neither be a single sentence nor even a collection of sentences and may contain information acquired from other sources (observation, knowledge base, beliefs, etc). We assume that the discourse utterances have already been partially processed from their original form so that they appear as expressions of the language Λ. This initial processing includes syntactic analysis in the categorial grammar CAT [6], [7], and the first stage translation into Λ by the Rule 1 (not reported here) which derives partial representations of sentences. For detail concerning text transformations not discussed here the reader is referred to [11] and [12].

2. Perfect Contexts

Let us consider the simple case of translating the following discourse fragment into predicate calculus representation.

Example 1

(1) *John interviewed a man.*

(2) *The bastard killed him.*

Suppose that the definite description of *the bastard* is intended to refer to the man mentioned in (1), and that the pronominal *him* refers to *John*. Assume also that (1) has the straightforward referential reading. To properly represent the meaning of (2), we first modify the traditional translation of the definite article *the*, [6], [8]. The new translation introduces the explicit reference to the context in which the definite description is instantiated. Thus we have

(3) **the** $\rightarrow$ $(\lambda P\,(\lambda C\,(\lambda Q\,(\exists x\,(P\,x)\ \&\ (C\,x)\ \&$
$(\forall y\,((P\,y)\ \&\ (C\,y))\supset (x{=}y))\ \&\ (Q\,x)))))$

Here the facts $(P\,x)$, $(Q\,x)$, and $(C\,x)$ give a characteristics of the referenced object. The part under the universal quantifier emphasizes the uniqueness of the x under the context C. Observe that the literal $(P\,y)$ is often insignificant for fixing a unique reference for the object (Example 1) The same cannot be said, however, of the instance $(P\,x)$ outside the scope of $\forall$. In this case we acquire some additional knowledge about the already selected individual. In the rest of this discussion we shall often drop the literal $(P\,y)$ whenever it does not lead to an ambiguous situation.

The first formal rule for translating two sentence paragraphs can be presented now. The rule is called the **Perfect Context**

Translation Rule because the context established by the first sentence in the pair has the perfect referential interpretation

---

RULE 2 (Perfect Context Translation Rule)†
An object u referenced by *the* has been mentioned previously in a *de re* context, thus its existence is presupposed. Let $S_1(u)$ be the context sentence which mentions u. Let $S_2(u)$ be the sentence in question. We have

(i) $S_1(\mathbf{a}\ P) \rightarrow (\exists u\ (P\ u)\ \&\ (F\ u))$

(ii) $S_2(\mathbf{the}\ P_1) \rightarrow (\exists u\ (C\ u)\ \&\ (\forall x\ ((P_1\ x)\ \&\ (C\ x)) \supset (x{=}u))\ \&\ (P_1\ u)\ \&\ (F_1\ u))$.

The context C is derived from S_1 as $(\lambda u\ (P\ u)\ \&\ (F\ u))$.

Using Rule 2, the Example 1 is translated as follows. Let (1) be as given before, i.e.,

(4) $1 \rightarrow (\exists x\ (m\ x)\ \&\ (i\ J\ x))$

We derive the context C from (1) as

(5) $(\lambda x\ (m\ x)\ \&\ (i\ J\ x))$

Then applying Rule 2 we derive the translation of (2) as

(6) $2 \rightarrow (\exists x\ (m\ x)\ \&\ (i\ J\ x)\ \&\ (\forall y\ ((b\ y)\ \&\ (m\ y)\ \&\ (i\ J\ y)) \supset (x{=}y))\ \&\ (b\ x)\ \&\ (k\ x\ J))$

where *m*, *i*, *b*, and *k* translate *man*, *interviewed*, *bastard*, and *killed*, respectively, and *John* is translated as $(\lambda P\ (P\ J))$. The formula in (6) says that the man whom John interviewed killed John. The Perfect Context Translation Rule explains some aspects of translating inter-sentential references but, as we shall see, it accounts for only the most straightforward referential situations which do not involve either imperfect or attitude report constructions.

3. Imperfect Contexts

We distinguish two very general classes of verbs found in natural language sentences. These are **imperfect verbs** like: *seek*, *want to* α, *go*, *build*, *imagine*, ... and **perfect verbs** like: *find*, *come*, *have*, *have written*, *have imagined* .... An informal definition of imperfect verbs follows.

Definition
A verb will be called *imperfect* if the immediate effects of the action or state described by this verb last at most as long as the action or state itself does, and its results on the surrounding world cannot be determined before the action or state is committed. □

The characteristics of imperfect verbs can be summarized as follows:

(a) They have no permanent influence on on the situation surrounding an utterance *want to marry*, *must have*, ...;

(b) They can be committed however, when turned into a perfect form: *have married*, *have*, ...;

(c) They (but not only they) can create non-referential translations of sentences;

(d) An imperfect verb v can be decomposed into an imperfect operator $\tilde{v}$, and the perfect form $\hat{v}$. Thus $\tilde{seek} = try$, $\hat{seek} = find$. ‡

(e) Complement taking imperfect verbs act as *imperfectness* operators on the complement, and its main verb, creating compound imperfect verbs. A special consideration will be given to the verbs *want* and *must*.

† We preserve the original rule numbering given in [12]. For the remaining rules the reader is referred there, and also to [11].

‡ In contrast with the classification of these verbs in the category (t/e)//(t/e) given in [6] we need imperfect operators in the category (t/e)/t, with the particle "to" being a part of the complement phrase.

Table 1 gives some examples of imperfect verbs and their perfect counterparts. It must be noted here that the perfect/imperfect distinction in interpretation of some verbs coincides, to some degree, with the notions of transparent and opaque readings of sentences containing such verbs [7]. However, the transparent/opaque distinction has been usually made in connection with the use of propositional attitudes, such as *want to* or *try to*, etc. [6] We are going somewhat further in our classification considering as imperfect operators not only propositional attitudes, but also all other syntactic and lexical properties of verbs (such as tense or mode) which may have similar effects on sentence reading. Consider only *A car may/will/must arrive* where the car will not materialize until it actually arrives.

Example 2
Consider the following new "story".

(7) *John wants to marry a queen.*

(8) *The queen must be wealthy.*

As the reader has perhaps already observed, the existence of the queen in (8) is not necessarily presupposed as a consequence of

Verb Forms		
imperfect	**perfect form**	**possible imperfect operator**
seek	find	try (to)
go	come	try (to)
go to α	α	go (to)
want to α	α	want (to)
wish to α	α	wish (to)
must α	α	must
be building	have built	try (to)

Table 1. Imperfect verbs and their perfect forms.

the possible *de dicto* reading of (7). Notice also that (8) would have completely different meaning when considered without the context supplied by (7). We can paraphrase (7) and (8) as (in a possible reading)

(9) *The queen John marries, if any, must be wealthy*

Two distinct reference situations can be observed. The situation where the context setting sentence has its referential reading (i.e. there exists a particular queen John wants to marry) is correctly represented by the Perfect Context Translation Rule. In such a case *wants to marry* behaves just as an ordinary perfect verb. This rule, however, cannot be used when both sentences have their non-referential readings. To account for non-referential readings in imperfect contexts we formulate a new rule called **Imperfect Context Translation Rule** given below.

RULE 3 (Imperfect Context Translation Rule)
An object u referenced by *the* has been recently mentioned in a *de dicto* environment, i.e its existence is not assumed. Let S_1, S_2 be defined as before. Then

(i) $S_1(\mathbf{a}\ P) \rightarrow (\mathbf{imp}\ (\exists u\ (P\ u)\ \&\ (F\ u)))$

where **imp** is the imperfect operator such that it is derived from the imperfect verb of S_1.

(ii) $S_2(\mathbf{the}\ P_1) \rightarrow (\mathbf{imp}_1\ (\exists u\ (C\ u)\ \&\ (\forall x\ ((P_1\ x)\ \&\ (C\ x)) \supset (x{=}u)))\ \&\ (P_1\ u)\ \&\ (F_1\ u)))$.

Where $\mathbf{imp}_1$ is the imperfect operator of S_2, and the context C is derived from S_1 as $(\lambda u\ (P\ u)\ \&\ (F\ u))$.

If a sentence with an imperfect verb has a referential reading in the form

(10) $(\exists x\,(P\,x)\ \&\ (F'\,x))$

then the non-referential reading featured in Rule 3 is obtained by realizing that **imp** $= \hat{F}'$, $F = F'$. Notice further that if we used a perfect verb in (8) as in

(11) *The queen is wealthy*

we would resolve the *de dicto/de re* ambiguity and **both** sentences would have only their *de re* readings. Thus the presence of an imperfect construction in (8) is essential for preserving (7)'s *de dicto* reading and for extending it over (8). It should also be clear that Rule 3 requires that the object referenced in the current utterance has non-referential status in an imperfect context. In other words, the passage

(12) *John married a queen.*
The queen must be wealthy.

has only a referential interpretation.

4. Attitude Report Contexts

As we saw, when a description is used consequently non-referentially the context setting situation changes so significantly that we need a separate rule to account for these cases. In Example 2 when *a queen* was used referentially we would further describe her as *the queen John wants to marry*, while when interpreting the "story" non-referentially we could only speak of *the queen John (eventually) marries (if any)* [6], [7], [2]. This is because in the non-referential reading the queen is not available for reference before John actually marries her. This difference has been properly accomodated by Rules 2 and 3.

We turn now to another class of verbs which can also create non-referential readings. These are **attitude report verbs** like *imagine that*, *see that*, *believe that*, etc. We restict our attention to the perfect contexts, in particular to those which do not involve any imperfect operators. The attitude report verbs used in such situations will be called **perfect attitude report verbs**. Although we are giving just one example with the attitude report verb *imagine that*,† the discussion below applies to other perfect attitude report verbs as well.

Example 3

Let us analyse the following "story" in detail.

(13) *John imagines that a unicorn lives in the park.*

(14) *The unicorn has a pink tail.*

Basically we can differentiate three reference situations between (13) and (14). As before we do not consider the trivial case where *a unicorn* of (13) and *the unicorn* of (14) do not co-refer. Assume first that we have following translation as given:

(15) *unicorn* $\rightarrow u$
imagines that $\rightarrow imt$
to have a pink tail $\rightarrow hpt$
to live in the park $\rightarrow lp$

CASE I. Suppose *a unicorn* in (13) is used referentially, that is, its existence is presupposed. We obtain (we consider here just one possible referential reading)

(16) 13 $\rightarrow$ $(\exists x\,(u\,x)\ \&\ (imt\ J\ (lp\,x)))$
14 $\rightarrow$ $(\exists x\,(u\,x)\ \&\ (C\,x)\ \&$
$(\forall y\,((u\,y)\ \&\ (C\,y)) \supset (x{=}y))\ \&\ (hpt\,x))$
where the context C is $(\lambda x\,(u\,x)\ \&\ (imt\ J\ (lp\,x)))$.

according to Perfect Context Translation Rule (Rule 2).

CASE II. Suppose to the contrary that *a unicorn* in (13) has been used non-referentially. That is

(17) 13 $\rightarrow$ $(imt\ J\ (\exists x\,(u\,x)\ \&\ (lp\,x)))$

We have now two options for translating (14). In one case *the unicorn* used in (14) refers to some particular individual the speaker of (14) knows but perhaps John does not. In this case

(18) 14 $\rightarrow$ $(\exists x\,(u\,x)\ \&\ (C\,x)\ \&$
$(\forall y\,((u\,y)\ \&\ (C\,y)) \supset (x{=}y))\ \&\ (hpt\,x))$
where $C = (\lambda x\,(u\,x)\ \&\ (imt\ J\ (lp\,x)))$ as before

This translation correctly emphasizes that from the point of view of the speaker of (14), *a unicorn* in (13) has been used referentially. Therefore **both** sentences get their referential translations and case **II** reduces to case **I**. Observe that case **II** is that of misunderstanding the intention of the speaker, but it is how the hearer interprets the discourse at the moment.

CASE III. On the other hand, let us assume that the speaker of (14) has the possibility of glancing into John's image of the unicorn and sees that it has a pink tail there. The speaker of (14) is therefore extending our information of what John imagines to:

(19) *John imagines that the unicorn he imagines to live in the park has a pink tail.*

Here, we should expect (14) to translate as

(20) 14 $\rightarrow$ $(imt\ J\ (\exists x\,(u\,x)\ \&\ (C\,x)\ \&$
$(\forall y\,((u\,y)\ \&\ (C\,y)) \supset (x{=}y))\ \&\ (hpt\,x)))$
where $C = (\lambda x\,(imt\ J\ ((u\,x)\ \&\ (lp\,x))))$

The speaker of (14) uses *the unicorn* **semi-referentially**, taking the image of unicorn as the context setting situation. He does not need any imperfect operator in his utterance. This is because the abstract situation created by an utterance of (13) persists for some (short) period of time after the utterance took place. This situation involves an implicit export of an attitude report operator from (13) into (14). That is, (14) should be understood now as

(21) *John imagines that the unicorn has a pink tail.*

but it is not necessary that the actants in (13) and (14) must co-refer. The attitude report verb *imagine that* although perfect (according to the definition given in section 3) has the ability to create non-referential readings, and then behave quite differently than an "ordinary" perfect verb (like those reported in section 2).

This discovery calls for yet another translation rule which we have chosen to name the **Attitude Report Context Translation Rule**.

RULE 4 (Attitude Report Context Translation Rule)
An object u referenced by *the* has been recently mentioned in a *de dicto* environment with an attitude report verb **att**. Let S_1, S_2 be defined as previously. Then the only non-referential reading of S_2 with respect to u can be obtained as

(i) $S_1(\mathbf{a}\ P) \rightarrow (\mathbf{att}\ (\exists u\,(P\,u)\ \&\ (F\,u)))$

(ii) $S_2(\mathbf{the}\ P_1) \rightarrow (\mathbf{att}_1\ (\exists u\,(C\,u)\ \&\ (\forall y\,((P_1\,y)\ \&\ (C\,y))$
$\supset (x{=}y))\ \&\ (P_1\,u)\ (F_1\,u)))$

The context C is derived from S_1 as $(\lambda u\ (\mathbf{att}\ ((P\,u)\ \&\ (F\,u))))$, and $\mathbf{att}_1$ is the implicit attitude report operator imported from S_1.

Thus far it appears that case **III** presents the only situation when a non-referential reading can be exported from an attitude report context. Montague [6] and Partee [7] seem to agree with that observation. Otherwise we would always find ourselves in case **II** unless, perhaps, the sentence S_2 contained an imperfect construction. Rule 4 can be easily generalized over all cases where $\mathbf{att}_1$ is an explicit attitude report construction in S_2 as in

(22) *John believes that a unicorn resembles Mary.*
He imagines that the animal has a single horn.

An important outcome of Rule 4 in its original formulation is that it can explain a part of what Donnellan [4] called the *attributive use* and Barwise and Perry [1] called *value free use* of definite descriptions.

† Although we classify "imagine that" as a perfect verb, its close relative "imagine" is a different verb, much like "create".

Example 4
Suppose someone says:

(23) *The man drinking the martini is a fool.*

in the sense that the definite description *the man drinking the martini* is used attributively [4]. For a speaker to utter (23) is to explore an implicit context-setting situation in which there is *a man drinking the martini*, if the definite description used in (23) is to be singular, not generic (see, for example [9]), as we assume here. Therefore, to say (23) is to make the reference in the following context.

(24) *I believe that there is a man drinking the martini.*

(25) *The man drinking the martini is a fool.*

The context-setting sentence of (24) is **implicit** here. A side-effect of this assumption is that one cannot use a pronoun in place of a definite description when saying (25). In general any attitude report verb **att** can be used in (24) and then imported to (25) according to Rule 4.

The three translation rules we presented thus far deal with reference situations created by the use of definite descriptions in discourse. Appropriate rules for translating pronominal reference cases can be readily formulated, and we will not investigate this problem here.

5. Conditional Contexts

In section 4 we found that a part of the so-called *attributive use* of some nominal expressions can be explained in terms of non-referential attitude report readings. In these cases we assumed that the speaker was addressing an entity whose existence was relative to his attitute toward it (beliefs, imaginations, etc). But this was just one side of the coin In the following example we list just a few sentences where attributive reading cannot be explained by attitudes.

Example 5
Apparently, the following sentences can be interpreted non-referentially without any reference to speaker attitudes.

(26) *The man who drinks the martini is a fool.*

(27) *The man who breaks the law will be prosecuted.*

(28) *A man who kills somebody is a murderer.* □

It should be relatively clear that the sentences like (26) to (28) above roughly fall under the following scheme.

(29) *A/The* α *such that* $P(him_k)$ F's

or, in other words

(30) **if** a/the α P's **then** he_k F's

Rewriting (30) to more formal notation we get

(31) if $(\exists x\ (\alpha\ x)\ \&\ (P\ x))$ **then** $(F\ him_k)$

Clearly (31) is just another way to say (29) if the latter is to be understood attributively. No such equivalence can be made when (29) is used referentially. The choice between *the* and *a* in (29) depends on the speaker confidence as to the uniqueness of the entity established in the condition part of the sentence. Observe that in (31) the part between **if** and **then** constitutes our context-setting utterance S_1, and the part past **then** is our S_2. It is not necessary to assume a pronominal reference between S_1 and S_2, as it has been suggested in (31). In fact the general form of a conditional context utterance may be taken as

(32) if S_1 (**a** P) **then** S_2 (**the** P_1).

Example 6
Consider the following pairs of sentences. If the first sentence in a pair is used non-referentially, it has an equivalent conditional context reading expressed by the second sentence in the pair.

(33) *The cat that Mary buys is a burmese.* [13]
If Mary buys a cat it is a burmese.

(34) *The man who kills somebody is a murderer.*
If a man kills somebody, he is a murderer

(35) *The man who breaks the law will be prosecuted.*
If a man breaks the law, he will be prosecuted.

Thus the first sentence in (33) gets its non-referential conditional context reading in the form:

(36) $(\exists x\ (cat\ x)\ \&\ (buys\ M\ x)) \supset (burmese\ x_n)$

Resolving pronominal refrence of x_n to x we are getting that

(37) $(\exists x\ (cat\ x)\ \&\ (buys\ M\ x)) \supset (\exists u\ (cat\ u)\ \&\ (buys\ M\ u)\ \&\ (\forall y\ ((cat\ y)\ \&\ (buys\ M\ y)) \supset (y{=}u))\ \&\ (burmese\ u))$

The example has been chosen so that both sides of the conditional context reading for (33) to (35) when taken alone have a straightforward perfect referential reading. This reading does not represent the general case, and other pronominal context translation rules may be useful for resolving pronominal references between the sides - consider only *If John wants to marry a queen, she must be wealthy*. As we will see, this reading is not equivalent to the imperfect reading of *The queen John wants to marry must be wealthy*.

We summarize our observations as a formal translation rule, using the notion of *conditional context pattern*, which roughly speaking has the form of $(\lambda S_1\ (\lambda S_2\ (S_1 \supset S_2))))$ as expressed in Λ.

RULE 10 (Conditional Context Translation Rule)
If a sentence S has a singular, referential over x, reading translating to

(i) $S(\textbf{the/a}\ P) \rightarrow (\exists x\ (P\ x)\ \&\ (U\ x)\ \&\ (Q\ x))$,

and every equivalent to it referential singular reading assumes the same form, where U is an optional uniqueness clause possibly present when *the* is used in S, i.e.,

(ii) $U = (\lambda x\ (C\ x)\ \&\ (\forall y\ ((P\ y)\ \&\ (C\ y)) \supset (x{=}y)))$

where C is an external context that *cannot* be instantiated, then the conditional context non-referential reading of S exists and is obtained as the result of the following derivation

(iii) $S(\textbf{the/a}\ P) \rightarrow [[L.\ Q].\ P]$†

where L is the conditional context pattern in the form

(iv) $L = (\lambda S_1\ (\lambda S_2\ ((\exists x\ (S_1\ x)) \supset (S_2\ x_n))))$

and the pronominal reference of x_n is resolved to the argument of S_1 by one of the pronominal context translation rules.

The condition (i) in the formulation of Rule 10 seems to be quite restrictive as to the form a sentence can take in order to qualify for conditional context non-referential reading. One can clearly see that the literal P can be verbalized as

(38) $(\lambda x\ (P_1\ x)\ such\ that\ (Q_1\ x))$

as for example in *a man who breaks the law*. Yet the restrictive wh-clause is not strictly necessary to maintain P in the requested form. If one says

(39) *A tiger is dangerous.*

a non-referential interpretation is still possible in conditional context with P = *an entity such that it is a tiger*, so that P translates as

(40) $P \rightarrow (\lambda x\ (E\ x)\ \&\ (tiger\ x))$

where E is the *entity* predicate which as non-significant may be omitted.

† Here [f, a] denotes functional application f.a.

The question naturally arises, what kind of sentences do not qualify for conditional context readings. Obviously, the utterance which pressuposes the existence does not qualify (but this very sentence does!). Compare

(41) *There is a unicorn with a pink tail which lives in the park.*

(42) *A unicorn which has a pink tail lives in the park.*

Apparently (41) has a strictly referential reading, and no non-referential reading is possible. Observe, however, that (41) has an equivalent referential reading in the form

(43) 41 $\rightarrow$ $(\exists x\ (u\text{-}with\text{-}pink\text{-}tail\text{-}which\text{-}lp\ x))$

which violates the restriction that *every* equivalent referential reading must meet the form of (i) from Rule 10. In contrast, (42) can be translated as

(44) 42 $\rightarrow$ $(\exists x\ (u\ x)\ \&\ (hpt\ x)\ \&\ (lp\ x))$
with $P = (\lambda x\ (u\ x)\ \&\ (hpt\ x))$ and $Q = (\lambda x\ (lp\ x))$.

The next observation to make in conjunction with Rule 10 is that the sentence S does not necessarily have to be a top level clause. Consider again the sentence

(45) *John wants to marry a queen.*

The referential reading of (45) assumes the already known to us form of

(46) $45ref \rightarrow (\exists x\ (q\ x)\ \&\ (w\ J\ (m\ J\ x)))$

Rule 10 immediately provides us with one possible conditional-context (perfect) reading as

(47) $45ccper \rightarrow (\exists x\ (q\ x)) \supset (w\ J\ (m\ J\ her_n))$

This translations reads somewhat like: *if there is a queen, John wants (to marry her)*, what constitutes a conditional reading of (45) which can be paraphrased as *The/An entity such that she is a queen [has the property that] John wants to marry her.* By Rule 3 we can also derive the imperfect reading of (45).

(48) $45imp \rightarrow (w\ J\ (\exists x\ (q\ x)\ \&\ (m\ J\ x)))$

Applying Rule 10 to the inner clause we get another conditional context (imperfect) reading of (45).

(49) $45ccimp \rightarrow (w\ J\ ((\exists x\ (q\ x)) \supset (m\ J\ her_n)))$

This reading can be paraphrased as *John wants [to achieve] that the/an entity such that she is a queen is married by him.*

There is a sublte difference in meaning between (45ccper) and (45ccimp). The former states that unless a queen exists John's attitude toward her cannot be instantiated, i.e., a queen's existence causes John's desire to become in effect. In the latter John wants to reach the state in which a queen's very existence will entail John's marrying her.

6. Conclusions

We presented a fragment of a new approach to representing the meaning content of natural language utterances in discourse. We examined four classes of in-text dependencies between utterances created by the use of definite descriptions, and formulated appropriate rules of translation into the lambda-categorial language Λ. We also suggested a new representation for the definite article *the* that makes explicit the context in which a definite description is instantiated. Careful examination of each context situation under investigation allowed for replacing intensional interpretations of certain types of utterances, as advocated by Montague [6], by more tractable representations which, we believe, will prove amenable for computer analysis. A number of related reference phenomena including pronominal references, referring to a name, as well as forward and indirect references (involving the hearer s knowledge base) can be solved within this framework

Finally, we have to stress that the presented solutions apply to the reference cases arising between singular nominal phrases and corresponding to them singular objects only i.e., to these which remain in the relation of relative singularity to one another We do not treat here the cases where an object (and its description) is classified as being (and as referring to) an instance of some more general object (such as an intension) with respect to some coordinate (such as the index). These latter objects and other similar concepts which we consider non-singular are discussed in detail in [12].

Acknowledgements

The author would like to thank Dr. Nick Cercone for his comments and suggestions that helped to improve the quality of this paper. This research was supported in part by the Natural Science and Engineering Research Council of Canada under Operating Grant number A4309, by the Office of the Academic Vice-President, Simon Fraser University, and by the SFU's Open Graduate Scholarship. Thank the LCCR for use of facilities.

References

[1] Barwise, J., J. Perry (1983). *Situations and Attitudes*. The MIT Press.

[2] Cooper, R. (1985). "Meaning Representation in Montague Grammar and Situation Semantics." Speech given at the TANLU Workshop, Halifax, Nova Scotia, 28-29 May, 1985.

[3] Cresswell, M. J. (1973). *Logics and Languages*. Methuen & Co.

[4] Donnellan, K. (1971). "Reference and Definite Descriptions." In D. D. Steinberg, L. A. Jakobovits (eds.), *Semantics*. Cambridge University Press. 100-114.

[5] Hirst, G. (1983) *Semantic Interpretation Against Ambiguity*. Doctoral Dissertation. Technical Report CS-83-25. Department of Computer Science, Brown University.

[6] Montague, R. (1974) "The Proper Treatment of Quantification in Ordinary English." In R. Thomason (1974). *Formal Philosophy*. Yale University Press.

[7] Partee, B. H. (1972). "Opacity, Coreference, and Pronouns." In D. Davison, G. Harman (eds.). *Semantics of Natural Language*. Reidel, Dordrecht. 415-441.

[8] Partee, B. H. (1976). "Some Transformational Extensions of Montague Grammar." In B. H. Partee (ed.), *Montague Grammar*. Academic Press. 51-76.

[9] Quine, W. V. (1960). *Word and Object*. The MIT Press. Cambridge, Mass

[10] Strzalkowski, T. (1984). "Toward a Proper Meaning Representation for Natural Languages." Working Paper 1. Natural Language Group, LCCR, Simon Fraser University. Burnaby, B.C.

[11] Strzalkowski, T., N. Cercone (1985). "A Framework for Computing Extra-Sentential References." Proceedings of the TANLU Workshop, Halifax, Nova Scotia. 107-116.

[12] Strzalkowski, T. (forthcoming). *A Theory of Stratified Meaning Representation*. Doctoral Dissertation. Department of Computing Science, Simon Fraser University. Burnaby, B.C.

[13] Webber, B. L. (1979). *A Formal Approach to Discourse Anaphora*. Doctoral Dissertation. Garland.

A DOMAIN-INDEPENDENT NATURAL LANGUAGE DATABASE INTERFACE

YAWAR ALI
RAYMOND AUBIN
BARRY HALL

BNR
Ottawa, Ontario, Canada

SOMMAIRE

Le développement d'une interface générique en langue naturelle a été entrepris pour un système de gestion de bases de données de type entité-relation. Au plan syntaxique, cette interface se fonde sur une grammaire syntagmatique augmentée qui conserve la commodité et l'efficacité d'une grammaire sémantique tout en éliminant sa trop grande spécificité. Au plan sémantique, la signification d'une requête est formée d'opérations générales de navigation sur la base de données. Ces opérations s'appliquent, en fait, à une base de données virtuelle, faite du schéma principal de la base additionné d'ensembles d'entités virtuels. En dernier essor, ces opérations sont traduites en termes de commandes auprès de la base de données réelle. Une telle interface sera d'autant plus pratique qu'elle sera dirigée vers une population d'usagers assortie à ses capacités.

ABSTRACT

An experimental domain-independent natural language interface has been developed for an existing entity-relationship database management system. Syntactically, it relies on an augmented phrase structure grammar which retains the convenience and efficiency of a semantic grammar while removing some of its ad hoc nature. Semantically, the interpretation of a query is in terms of general database navigation operations. These operations apply to a virtual database, made of the actual database schema augmented with derived entity sets. Navigation operations are eventually translated to actual database commands. Such an interface will be practical inasmuch as it is aimed at the right population of users.

INTRODUCTION

Facilitating access to services and information is a key objective of computer systems. Therefore, it is not surprising that one of the first real natural language processing systems, LUNAR (Woods, Kaplan, and Nash-Webber 1973), was developed for database access. This successful prototype stimulated research in natural language interface systems which eventually led to a first commercial product, INTELLECT (Harris 1979). More recent systems have aimed at developing tools for portability, e.g., Grosz (1983).

SESAME is an experimental natural language system developed at BNR for interfacing to a proprietary entity-relationship database management system. The objectives of this work are domain-independent architecture, adequate coverage and resilience, and efficiency. Testing the interface on a real database with users has also been an important goal. The Northern Telecom Customer Service Report database was selected; it is used in the field support of the DMS[1] family of digital switches.

The goal of domain-independence is realized in the two key complementary components of SESAME: parsing and interpretation. First, SESAME uses an augmented phrase structure grammar which retains the convenience and efficiency of a semantic grammar (Burton 1976) while removing most of its ad hoc nature and hence, improving its portability. More precisely, it is a syntactic grammar augmented with so-called semantic variables. The parser uses these variables to ensure the semantic correctness of a query. Semantic variables are only instantiated by a domain-dependent lexicon.

Second, the interpretation of a query is in terms of general database navigation operations applied to domain-dependent parameters obtained from the lexicon. These operations are built compositionally as a query is parsed, and then globally simplified. In its simplest form, a database navigation operation is made of a Select and a list of Join and Project, called a Path. The PRE system (Epstein 1985) generates queries following a similar pattern. SESAME, however, goes further. If needed, it can intersect paths. Furthermore, it operates over a virtual database, made of the actual database schema augmented with derived entity sets. This augmented schema is defined in a separate domain-dependent component and achieves a form of mediation.

---

[1] Trademark of Northern Telecom Ltd.

The architecture of SESAME is thus characterized by separate domain-dependent and domain-independent modules. The complete structure is pictured in Figure 1. To move from one domain to another, only the synonym and main lexicons have to be replaced in addition to the database and its augmented schema.

Like the LADDER system (Hendrix et al. 1978), SESAME has separate, self-contained modules for linguistic analysis and intelligent data access.[2] Both systems provide their users with virtual views of the data. However, unlike the semantic grammar utilized by LADDER, SESAME's grammar is completely domain-independent and portable.

The next section will describe the linguistic part of SESAME with details on the grammar formalism and its application to the handling of elliptical inputs. The subsequent section will describe the generation of navigation operations and their translation into actual commands for the target database. The paper will conclude with some lessons learned from SESAME and the perceived role of natural language interfaces to databases.

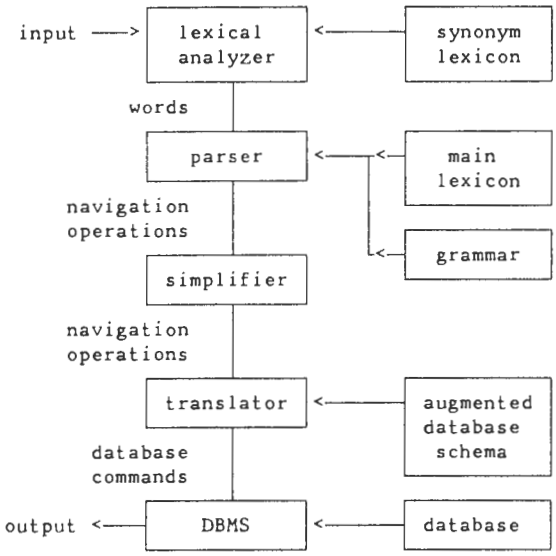


Figure 1. SESAME architecture

LINGUISTIC ASPECTS

The linguistic component of SESAME takes natural language input typed in by the user and produces as output an intermediate language expression constituting the interpretation of the user's query.

[2] Unlike LADDER, SESAME does not support access to distributed data.

The translation proceeds in a number of phases, including lexical analysis, parsing, and possibly, processing by specialized ellipsis and pronoun resolution algorithms. We now discuss the major phases of linguistic analysis.

Parsing

Initially, the user's input is subjected to lexical analysis, during which substitutions are made for pre-defined synonyms, and an attempt is made to correct misspelt words using a spelling corrector.

The preprocessed input is passed on to the parser, which makes use of a domain-independent grammar and two lexicons, one domain-independent and one domain-specific. The grammar is written using an augmented phrase structure formalism of our own design. This formalism allows context-free phrase structure rules to be augmented with variables that can be bound to the semantic categories of subconstituents. During parsing, these semantic variables are matched against one another in order to filter out inputs that are syntactically well formed but semantically unacceptable. Associated with each grammar rule is a template for an intermediate language expression, representing the interpretation of the phrase parsed by the grammar rule. During parsing, the intermediate language expressions associated with applicable grammar rules are composed, resulting in the intermediate language interpretation of the user's query, which is then passed on to the simplifier.

Using this formalism, we are able to construct grammars that generate fewer ambiguities than do conventional syntactic grammars, but which are much more compact than the semantic grammars used by earlier systems, such as LADDER (Hendrix et al. 1978). As an illustration of this point, consider the commonly occurring case of inputs which include a noun followed by two prepositional phrases, as in:

1. managers of departments with graduate degrees

2. locations of customers with software problems

3. salaries of employees with sales background

From a syntactic point of view, such inputs are ambiguous, depending on which noun the second prepositional phrase is taken to modify. Thus, a purely syntactic grammar might use a few recursive rules to parse each of the sample inputs shown above, as in Figure 2, yet would produce two parses for each input. For instance, in the case of (1), one parse would associate "graduate degrees" with "managers" whereas the other would associate "graduate degrees" with "departments".

On the other hand, a semantic grammar would use type information to rule out the second, semantically unacceptable interpretation, resulting in

```
<NP> ::= <Noun>
<NP> ::= <Noun> <PP>
<NP> ::= <Noun> <PP> <PP>
<PP> ::= <Preposition> <NP>
```

Figure 2. Fragment of a syntactic grammar

a single parse. However, the semantic grammar would require distinct grammar rules to parse (2) and (3), as shown in Figure 3, since they mention entities from different semantic categories, even though the syntactic structure of the two phrases is the same.

In contrast, the SESAME grammar combines the advantages of both syntactic and semantic grammars, making use of general lexical categories, such as noun and verb, in order to avoid the proliferation of grammar rules that is characteristic of semantic grammars, and using semantic variable matching to avoid producing as many parses as would syntactic grammars. Thus, SESAME can parse inputs like (1), (2) , and (3) using as few grammar rules as a syntactic grammar (see Figure 4)[3]. However, where a syntactic grammar would generate two parses for each of these inputs, SESAME, like a semantic grammar, would only produce a single, semantically valid parse for each of the inputs.

Our grammar is domain-independent. In addition to a small domain-independent lexicon, it utilizes a domain-specific lexicon which lists meaningful

```
<OBJ> ::= <Mgr> <MgrQual> <MgrQual>
<OBJ> ::= <Loc> <LocQual>
<OBJ> ::= <Sal> <SalQual>
<Mgr> ::= managers
<Loc> ::= locations
<Sal> ::= salaries
<MgrQual> ::= of departments
<MgrQual> ::= with graduate degrees
<LocQual> ::= of <Cus>
<SalQual> ::= of <Emp>
<Cus> ::= customers <CusQual>
<Emp> ::= employees <EmpQual>
<CusQual> ::= with software problems
<EmpQual> ::= with sales background
```

Figure 3. Fragment of a semantic grammar

terms in the application domain and assigns values to semantic variables.

Thus, we achieve portability, while retaining the efficiency of a semantic grammar. Moreover, since we restrict our attention to task-oriented sublanguages, we do not require, or provide, complicated mechanisms to handle the intricacies of unconstrained English. As a result, our grammars are easier to construct and are more transparent than those written in formalisms such as Augmented Transition Networks (Woods 1970) or Lexical-functional Grammar (Bresnan and Kaplan 1982).

Parsing is carried out in a top-down, left-to-right manner, using an efficient iterative algorithm. A well-formed substring table is used to avoid reparsing previously analyzed subconstituents. Parse trees constructed by the parser are saved on a context stack for use by the ellipsis and pronoun resolution algorithms.

```
<NP:x> ::= <Noun:x>
<NP:x> ::= <Noun:x> <PP:x>
<NP:x> ::= <Noun:x> <PP:x> <PP:x>
<PP:x> ::= <Preposition:x:y> <NP:y>
```

Figure 4. Fragment of SESAME's grammar

Ellipsis and Pronoun Resolution

The ellipsis mechanism is invoked if an input fails to parse normally. This mechanism attempts to parse recognizable subphrases in the input, creating a list of fragments, each being a partial parse tree, or in the worst case an individual word. This list of elliptic fragments is then matched against the saved parse trees of previous inputs, starting with the most recent one.

If all fragments are syntactically matched, they are tested for semantic variable compatibility. In case of any semantic incompatibility, the algorithm backtracks and the search continues. If all semantic tests succeed, a new parse tree is constructed from the matched one by substituting the elliptic fragments for their matched components.

Pronoun resolution, similarly, takes advantage of the fact that both syntactic and semantic information is saved in parse trees. The presence of a pronoun in the input triggers a search for compatible referents in old parse trees. Once a match has been found, the subtree corresponding to the matched constituent is copied from the old parse tree into the new one, replacing the dummy node originally associated with the pronoun.

DATABASE NAVIGATION

Database navigation operations are expressed in an intermediate language that is modelled on Relational Algebra, though not all algebraic operators are provided. The operators that we do provide are as follows:

1. SEL:

[3] In the sample SESAME grammar rules shown in Figure 4, semantic variable names are associated with non-terminals. The name of a semantic variable is shown separated from the name of its associated non-terminal by a colon.

- corresponds to selection
- takes as argument the name of an entity set, optionally with qualifiers

2. INTER:

- corresponds to intersection
- takes as arguments one or more entity sets

3. YFER:

- corresponds to a join followed by projection
- takes as arguments an entity set and the name of a relationship between the first argument and another entity set
- effect is to perform a join between the supplied entity set and the entity set linked to it by the specified relationship, projecting out the fields of the latter entity set

Generation of Navigation Operations

Intermediate language expressions are built during parsing by filling in and composing templates associated with individual grammar rules. Thus, parsing of a simple noun phrase generates a SEL operation; e.g., parsing "Ottawa" might generate

```
(SEL CITY (EQ CITY_NAME OTTAWA)).
```

A noun qualified by a prepositional phrase corresponds to a SEL followed by a YFER, e.g., "staff in Ottawa" might translate to

```
(YFER (RCITYSTAFFLOC)
      (SEL CITY (EQ CITY_NAME OTTAWA))),
```

which specifies selection of all CITY records with "Ottawa" as value of the CITY_NAME field, followed by a join with the set of STAFF records through the RCITYSTAFFLOC relationship, projecting out the fields of the latter entity set.

As a further illustration, parsing a noun phrase with two qualifying phrases would correspond to an INTER of two expressions, each consisting of a SEL followed by a YFER, e.g., parsing "staff in Ottawa managed by Smith" might yield

```
(INTER (YFER (RCITYSTAFFLOC)
             (SEL CITY
                  (EQ CITY_NAME OTTAWA)))
       (YFER (RMANAGERSTAFFMGR)
             (SEL MANAGER
                  (EQ LAST_NAME SMITH))))
```

where RMANAGERSTAFFMGR is the name of a relationship between the MANAGER and STAFF entity sets.

Generation of Database Commands

Simplified navigation operations are passed on to the translator, which generates the corresponding sequence of commands in the target database language. In order to do this, the translator needs access to the database schema, which is made available to it in a file. The translation of simple database navigation operations into the target database language is straightforward, and the translator possesses rules for translating each different type of navigation operation. Complex, nested intermediate language expressions are recursively decomposed until their constituents are simple enough to be directly translated. Once the translations of the subconstituents have been generated and written out to a file, the translations of their embedding expressions can be generated, allowing references to entity sets derived while translating the subconstituents. When the translation is complete, the file containing the database commands is passed on to the database management system, which processes it and responds to the user.

Virtual Entity Sets and the Augmented Schema

The database schema made available to the translator can be augmented with virtual entity sets in order to provide a form of mediation between the user's view of the domain and the actual organization of the database. This permits the user to refer to a virtual entity set in the same manner as an actual entity set. Moreover, since the grammar is domain-independent, it does not distinguish between actual and virtual entity sets when generating navigation operations. During the database command generation phase, the translator replaces references to virtual entity sets by sequences of operations on actual entity sets, as specified by their definitions in the augmented schema.

For example, in our current application domain, users make use of the concepts of departments, staff, and managers. However, while the database schema defines the entity sets DEPARTMENT and STAFF, there is no entity set corresponding to the concept of a manager. Rather, the DEPARTMENT entity set has fields for the manager's last name and initials. SESAME allows the user to remain unaware of the details of the representation, being able to refer to managers, departments, or staff using the same syntactic constructs. During parsing, occurrences of the noun "manager" will translate to

```
(SEL MANAGER)
```

in the intermediate representation, just as "department" would translate to

```
(SEL DEPARTMENT).
```

However, during command generation the translator would make use of the definition of MANAGER in the augmented schema to translate

(SEL MANAGER)

to selection of the DEPARTMENT entity set and projection of the fields corresponding to the manager's last name and initials.

CONCLUSION

SESAME has met its main objectives. Architectural independence from the database domain and schema was achieved through (1) the separation between a domain-independent syntactic grammar and a domain-dependent lexicon and (2) the separation between general database navigation operations and the actual database schema augmented with derived entity sets. Efficiency was obtained by using an augmented phrase structure grammar formalism interleaving syntactic and semantic processing. A reasonable effort was made to give feedback to the user upon failure of the interface by displaying the unrecognized input fragments.

The grammar currently contains rules for parsing moderately complex inputs, including instances of WH-Questions, simple and nested relative clauses, and some conjunctions. Getting adequate coverage has remained an elusive goal, however. In fact, no universal notion of coverage could be found. Four classes of users were identified for the Customer Service Report database with different usage patterns: (1) clerks handle a large number of simple routine queries, (2) administrators track particular problems, (3) managers ask for trends, and (4) customers call parameterized prepackaged reports.

A purported advantage of natural language interfaces is that one can express one's thoughts immediately. This is a mixed blessing for the managerial type of usage and can create false expectations. Indeed, managers think in terms fairly remote from the data and would need an expert decision support system tied to the database. This is beyond the reach of SESAME and of most real natural language systems today. At the other extreme, a natural language interface like SESAME is probably an overkill for the usage customers make of the database.

Clerks and administrators represent the best target users. They have a good semantic knowledge of the domain and, most of the time, know what information they want. They can benefit from a natural language interface because not all are experts in the database command language syntax and because natural language requires so much less typing.

The capabilities and limitations of a natural language interface are best assessed with respect to a target population of users. So far, a small sample of people have experimented with SESAME; plans are being made to conduct a more comprehensive trial.

REFERENCES.

1. J.W. Bresnan, and R.M. Kaplan, "Lexical-functional grammar; a formal system for grammatical representation". In: The Mental Representation of Grammatical Relations. Edited by J.W. Bresnan. MIT Press, Cambridge, Mass., 1982.

2. R.R. Burton, Semantic Grammar: an Engineering Technique for Constructing Natural Language Understanding Systems, BBN Report No. 3453, Bolt Beranek and Newman, Cambridge, Mass. (1976).

3. S.S. Epstein, "Transportable Natural Language Processing through Simplicity - The PRE System" *ACM Transactions on Office Information Systems 3*(2) (April, 1985), pp. 107-120.

4. B.J. Grosz, "TEAM: a Transportable Natural-Language Interface System" In *Proceedings of the Conference on Applied Natural Language Processing*, Santa Monica, CA (February, 1983), pp. 39-45.

5. L. Harris, "Experience with ROBOT in Twelve Commercial Natural Language Data Base Query Applications" In *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, Tokyo (1979), pp. 365-368.

6. G.H. Hendrix, E.D. Sacerdoti, D. Sagalowicz, and J. Slocum, "Developing a Natural Language Interface to Complex Data", *ACM Transactions on Database Systems 3*(2) (1978), pp. 105-147.

7. W.A. Woods, "Transition Network Grammars for Natural Language Analysis", *Communications of the ACM 13* (1970), pp. 591-606.

8. W.A. Woods, R.M. Kaplan, and B. Nash-Webber, "The Lunar Sciences Natural Language Information System" In *Proceedings of AFIP Conference 42*, Montvale, NJ (1973), pp. 441-450.

NATURAL LANGUAGE REPORT SYNTHESIS :
An Application to Marine Weather Forecasts

R. Kittredge and A. Polguère
Département de Linguistique
Université de Montréal

E. Goldberg
Atmospheric Environment Service
Environment Canada, Toronto

Abstract

Certain varieties of text serve primarily to summarize and communicate formatted data. Such texts can often be synthesized directly from data without recourse to general planning mechanisms by use of a detailed grammar which captures the constraints on words, sentences and texts which are natural to the domain. A modular system is presented which synthesizes Arctic marine forecasts, drawing on domain-specific linguistic and non-linguistic knowledge. Among the problems faced were the linguistic treatment of data salience relations and the modulation of temporal adverbs to reflect levels of certainty for remote events. The present work can easily be extended to the synthesis of bilingual or multilingual reports.

1. Natural Language Report Synthesis

We use the term "natural language report synthesis" (NLRS) to describe the process of creating well-formed text which summarizes formatted data in a given domain using a style which mirrors the conventions of professional report writers for that domain.

NLRS for highly restricted domains was first demonstrated in the work of Kukich [1] on "knowledge-based generation" of stock market reports. Kukich's ANA system produces professional-sounding stock market summaries using a daily trace of Dow Jones' half-hourly quotations for the market average and major indices. Both ANA and the analogous FRANA system for French [2] have used a phrasal lexicon approach [3] which limits the generality of the linguistic component, but which seems to suffice for small and stereotyped domains.

NLRS differs from most work on text generation by its lack of a full-fledged planning mechanism for the organization of text structure. Instead, NLRS relies on a detailed grammar of text and sentence structure, augmented by a domain-specific lexicon, to capture the natural tendencies of professional report writers. Within stereotyped domains such as weather forecasting, the global structural choices tend to be more limited than, for example, in legal argumentation where powerful planning approaches are required [4]. The work described here follows the NLRS approach, but with a more modular organization. Its approach to lexical "insertion" is based on linguistically motivated categories and semantic classes, making transparent the relations between lexical items. There is relatively little use of phrasal strings compared with earlier NLRS work. Work on RAREAS also tests a new application domain which has raised a general new problem for temporal reference (cf. section 7).

2. Synthesis of Arctic Marine Weather Forecasts

The RAREAS system was developed during a five-month effort to explore the feasibility of synthesizing marine weather bulletins from formatted weather forecast data. The particular task was to produce Arctic marine forecasts for five forecast areas to the east of Baffin Island (known as FPCN25 forecasts). Marine forecasts are one of several types of weather bulletin based on the same basic weather data, each type emphasizing the conditions of interest to a particular community of users. In the case of marine bulletins, linguistic emphasis is placed on wind direction and speed, dangerous wind and freezing spray conditions, etc. RAREAS is designed to be sufficiently modular and flexible so as to allow easy extension and adaptation to other types of weather bulletin (e.g., agricultural bulletins, public weather forecasts). Although the current project seems to have proved the feasibility of automatically synthesizing weather forecasts, extensive testing and refinement are required before RAREAS or any successor can be introduced into daily use.

The RAREAS system is the natural language component of the MARWORDS project, which envisages automating the process of creating bulletins from meteorological information. In the current manual procedure all the available meteorological information (observations, radar and satellite imagery, and numerical weather prediction products) is made available to the weather forecaster. The weather forecaster must correctly diagnose the meteorological processes which will affect his particular area of interest throughout the forecast period, and then translate this knowledge into appropriate textual forecasts for various users.

In the proposed automated process, MARWORDS will use predicted values for meteorological parameters such as wind speed and direction, cloud cover, and others. In some cases, these predictions could be obtained directly from numerical weather prediction products. In most cases though, they would still be the result of a manual (i.e., human) forecasting procedure. MARWORDS will significantly reduce the workload on the forecaster, making it possible to focus more attention on meteorological problems.

In the normal course of events, the predicted values make up a continuum in both time and space. For simplicity, values are often given at regular steps in time (e.g., hourly) and space (either at grid points, or at weather observing sites). Alternatively, forecast parameters may be given in terms of significant changes only. MARWORDS is flexible enough to accept both types of data description. In fact, the structure and nature of the required data is a problem which needs more work to resolve.

3. Design of the RAREAS system

A major task in designing RAREAS was the definition of an input data format which properly divides the work between the MARWORDS expert system, which computes predicted values of weather parameters based on large-scale observations, and RAREAS itself, which interprets that data under local conditions for the purpose of marine forecasts. The format and its permissible content should have sufficient expressive power to reflect the nuances found in natural language forecasts. Ideally, the expert

system should be kept as independent of forecast purpose as possible. RAREAS should therefore take care of all matters related to subjective evaluation of the data (e.g., importance of the individual parameters for marine forecasts), as well as the linguistic expression of data values and data relations.

In its current form (see figure 1) RAREAS is a set of MProlog programs, which call each other (solid lines) to control the information flow (dotted lines) from data through to text.

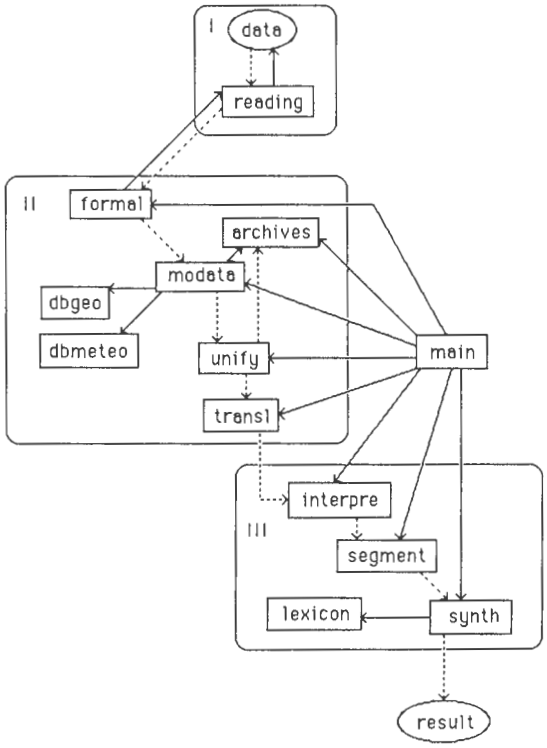


Figure 1. RAREAS as set of linked MProlog programs

These programs may be grouped conceptually into three components. Component I reads and parses formatted data from the input file, converting it into a regularized set of formulas directly evaluable by Prolog.

Component II carries out a number of non-linguistic operations prior to linguistic synthesis. In particular, data is checked for consistency and plausibility (program "modata") using databases of geographical and meteorological information ("dbgeo" and "dbmeteo"). Default values for less critical parameters may be inserted at this point when they are absent in the input data. Modata checks for conditions which are hazardous for marine operations (e.g., freezing spray, calculated as a function of forecast wind speed and air temperature, and of a seasonally and regionally adjusted water temperature taken from the database). Modata also consults an "archive" of data from the preceding forecast to formulate any necessary warnings as a function of those which may already be in effect.

One of RAREAS most difficult tasks is to merge or "unify" forecasts for contiguous areas when their forecast data is identical or nearly so; when similarity threshold conditions are satisfied a single report formula is created for the merged areas under a header which lists those areas.

Following unification, data not sufficiently salient for explicit inclusion in the report may be suppressed. For example, temperature is generally dropped after its use to check if freezing spray conditions are present. The resulting formulas are then translated into pre-linguistic "logical" representation (program "transl").

In component III all properly linguistic operations are carried out, including:

- interpretation of transitions between weather events into the same logical form as is used for event descriptions;
- segmentation of resulting logical structures into more independent pre-linguistic clauses and sentences;
- synthesis of textual form, instantiating sets of semantic constraints as individual lexical items, by consulting a "lexicon" module.

4. A Sample Report

The following simplified example (figure 2) shows the input formatted data, using mnemonic descriptors, for a sample subset of Arctic marine areas.

```
2200 mon 83/09/22 end.
frob wind 220 30 &
              nt 5 300 35 & nt 18 speed 40
              wea rain cont heavy &
                 nt 15 nl n 65 rain per moderate
              temp -3
              end.
cumb wind 100 25 &
              nt 3 nl n 60 200 35
              sky ovc & nt 13 bkn
              wea rain per & nt 10 stop & fog per
              temp 0
              end.
davi wind 100 25 &
              nt 3 nl n 60 205 35
              sky ovc
              wea snow per &
                 fog
              temp 0
              end.
              $
```

Figure 2. Sample RAREAS formatted input.

The formatted data identifies the Greenwich time of report validity, the date and area concerned, and then specifies initial values for each important weather parameter. Subsequent changes in the value of a parameter are preceded by the number of hours until the forecast change. Localized exceptions to the general forecast are preceded by a coded sub-area specification. At present, input data is limited to the six most important parameters: (1) wind direction, (2) wind speed, (3) cloud cover classification, (4) precipitation types (if any), (5) precipitation frequency and intensity rating, and (6) air temperature. Further forecast parameters which are functions of the input parameters (e.g., warnings and visibility ratings) are calculated by the first non-linguistic module.

After reading and analysis, the data is manipulated in clausal form through data checking, area unification and data suppression stages mentioned above. It is then translated into a "logical form" just before input to the linguistic modules.

Linguistic modules first calculate the values of significant semantic features of incipient lexical items, particularly regarding direction and degree of changes. For example, winds which

change direction in a clockwise direction will be described lexically as "veering" to the new direction, whereas winds which change in a counterclockwise direction are described as "backing". Initial lexical instantiation uses the most precise term available in the lexicon. Subsequent segmentation into sentences may juxtapose clauses in such a way that lexical variation is desirable. Precise terms may then be replaced by synonymic variants, or by more general (hyperonymic) lexemes.

Figure 3 gives the final textual form of the marine forecast corresponding to the data of figure 2 above.

MARINE FORECASTS FOR ARCTIC WATERS ISSUED BY ENVIRONMENT CANADA AT 3:00 PM MDT MONDAY SEPTEMBER 22 1983 FOR TONIGHT AND TUESDAY.

FROBISHER-BAY
GALE WARNING ISSUED ...
FREEZING-SPRAY WARNING ISSUED ...
WINDS SOUTHWESTERLY 30 VEERING AND STRENGTHENING TO NORTHWESTERLY GALES 35 LATE THIS EVENING THEN STRENGTHENING TO NORTHWESTERLY GALES 40 LATE TUESDAY AFTERNOON. CLOUDY WITH RAIN THEN SHOWERS DEVELOPING NORTH OF 65 N LATITUDE TUESDAY. VISIBILITY FAIR IN PRECIPITATION.

CUMBERLAND SOUND
GALE WARNING ISSUED ...
WINDS EASTERLY 25 VEERING AND STRENGTHENING TO SOUTHERLY GALES 35 NORTH OF 60 N LATITUDE THIS EVENING. CLOUDY WITH SHOWERS ENDING TONIGHT. FOG PATCHES TUESDAY. VISIBILITY FAIR IN SHOWERS AND POOR IN FOG.

DAVIS STRAIT
GALE WARNING ISSUED ...
WINDS EASTERLY 25 VEERING AND STRENGTHENING TO SOUTHWESTERLY GALES 35 NORTH OF 60 N LATITUDE THIS EVENING. CLOUDY WITH FLURRIES. FOG PATCHES. VISIBILITY FAIR IN FLURRIES AND POOR IN FOG.

Figure 3. RAREAS output for data of fig. 2 above.

5.1 Knowledge Sources for Report Synthesis

The RAREAS architecture isolates different types of linguistic and non-linguistic knowledge within appropriate modules. Our grammatical, lexical, rhetorical and stylistic description is based on an examination of all the marine bulletins (manually) produced for the FPCN25 region during the 1983 and 1985 seasons (some 50,000 words in all).

Examination of this extensive corpus of English has led to a fairly detailed grammar of this sublanguage (cf. [5],[6]).

Linguistic knowledge is broken down into several types:

- lexical semantics, including conditions for appropriate usage of words as a function of semantic configurations, particular data values, and word class co-occurrence restrictions;
- frequency preferences among synonymous terms in the sublanguage of marine bulletins;
- syntactic patterns, including the possible and preferred sentence patterns for expressing messages of given types; a second type of syntactic knowledge concerns the rules for deleting repeated sentence constituents when two or more propositions are fused into a single report sentence;
- simple principles of text organization, specific to the variety of text to be synthesized, and hence a function of the data salience hierarchy (see below);

Non-linguistic knowledge is of three types:
- geographical knowledge for each forecast area including (1) its time zone, (2) its limits of latitude and longitude, and (3) the names of adjoining areas (to allow recursively merging adjacent areas in case of similar meteorological regimes);
- meteorological data including (1) mean temperature values for air and water during each month of the Arctic shipping season (June through October) and (2) record values for temperature & wind speed;
- an "archive" of data from preceding reports, used to verify if dangerous wind warnings or freezing spray warnings are in effect.

Geographic knowledge is used primarily during the attempt to merge reports for adjoining areas. However time zone data is used to calculate local time associated with meteorological phenomena, and hence allow attribution of appropriate temporal descriptors (e.g., "by late afternoon"). Input data to the system has only the Greenwich reference time used by meteorologists.

6. Linguistic Treatment of Salience

The structure of marine weather forecasts shows several linguistic correlates of data salience relations. First, warnings of dangerous conditions (strong winds and freezing spray in the FPCN25 region) constitute separate headers preceding the normal text. Only warnings are so positionally marked and informationally redundant. Within the normal text, sentence groups dealing with each forecast parameter are ordered by two principles: intrinsic interest of the data and implicit causal links between the events or states described. Thus wind direction and speed, as the critical factors in marine conditions, occupy initial position. However visibility ratings, which should follow in order of importance, occur last by virtue of their dependence on fog/mist descriptions, which in turn are somewhat dependent on precipitation, which in turn follow cloud cover ratings. Sentence groups are therefore ordered as follows:

WINDS > CLOUD-COVER > PRECIP >
FOG&MIST > VISIBILITY

Within each sentence group, sentences and clauses are first ordered according to the dichotomy "general vs. local exception", and then chronologically within general and exceptional parts. A final correlate of data salience is the choice of marked lexical items and modifiers. For example, particularly strong winds are classified as "gales" (at 35 knots), "storm force winds" (at 45 knots), etc. Also, more specialized sense verbs such as "veering" and "backing" tend to be used more for large changes of wind direction.

7. Temporal Reference under Increasing Uncertainty

An interesting problem arises in ascribing particular time adverbials to points and intervals of (local) time. There appears to be a tendency in reports to "hedge" temporal descriptors slightly as reference time becomes more remote from the forecast issue time. For example, "Tuesday afternoon" or "by (Tuesday) evening" may be preferred for remote reference over the more precise "late Tuesday afternoon". This may reflect the increasing difficulty in predicting onset times for remote meteorological events. RAREAS synthesizes time adverbials for events as a function of two factors: (1) the remoteness of the event (within or beyond 24 hours from forecast issue time), and (2) whether the event is essentially a point

or an interval of time. In the case of time intervals, a more precise interval description can be chosen if all points in the interval fall within a day subpart (e.g., "Wednesday morning"), than if the interval straddles two day subparts, requiring a hyperonymic adverbial (e.g., "Wednesday").

8. Bilingual Reports

The RAREAS system was designed to accomodate the synthesis of marine weather bulletins in French as well as in English. Only the final three components in the processing sequence are language-dependent (and only the last of these in a non-trivial way). Syntactic patterns and lexical entries for French must of course be furnished on the basis of independent linguistic study of the corresponding French sublanguage. The exact semantics for French (correspondences between data configurations and specific lexemes) must be worked out separately, since there is no guarantee that English and French are lexically one-to-one, even in this narrow domain.

Canadian weather forecasts of all varieties are currently translated into French by the METEO system [7], developed at the Université de Montréal some ten years ago. Although METEO takes advantage of the relative closure and stereotyped style of forecasts, a certain percentage of forecast sentences fails analysis and hence translation. This is due not only to input errors due to typing and line noise, but also to slight irregularities in the usage of English grammar and lexicon on the part of forecasters, which have proved troublesome to foresee in a compact system.

The automatic synthesis of marine forecasts, on the other hand, should eliminate the fuzzy edges of unpredictability in human language production, by using a semantically complete and consistent subset of language to cover all foreseeable data configurations. The choice of this subset is a matter of linguistic engineering, but is basically determined by the relative frequencies of usage of synonymic and paraphrastic alternatives. One tries to favor most frequent choices among the alternatives, while minimizing the semantic overlap and guaranteeing full coverage of all "reasonably sayable" contents. In short, one strives for a kind of "minimal covering" which preserves some symmetry or systematicity within the linguistic means used. This normally involves "legislating" a clearcut semantics for each verb to replace the collective tendencies of usage of meteorologists. This does not exclude reintroducing lexical and syntactic variation (to give a more natural style) as the rule-based avoidance of repetition at the level of words and sentence structures. Repetition avoidance can then be governed by a process of substituting more general meanings for more specific ones when exact synonyms are not available.

Work on RAREAS may be seen as preparing the ground for an attractive alternative to machine translation of weather forecasts. The simultaneous synthesis of English and French forecasts directly from data would optimize the transfer of information to speakers of both languages. Parallel synthesis of bilingual forecasts bypasses translation altogether and minimizes human intervention, thus maximizing speed of transfer and (in principle) reliability. RAREAS' logical structures for English forecasts are probably close to what is needed for French. Most of the system's work with a particular set of input data would therefore serve towards the synthesis of a report in either language.

9. Implementation

RAREAS is written in MProlog, and runs on a Vax under VMS as well as on PC/XT/AT compatible microcomputers. Synthesis of a complete five-area forecast (about 150 words) takes about half a minute for the Vax implementation and a minute for the AT implementation. In either case, this is probably less than the time required to type or write the same forecast by hand, not to mention compose the same forecast from data.

10. References

[1] K. Kukich (1983) "Design of a Knowledge-Based Report Generator", *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics.*

[2] C. Contant (1986) "Génération automatique de texte: application au sous-langage boursier", M.A. thesis, Dépt. de Linguistique, Université de Montréal.

[3] J. Becker (1975) "The Phrasal Lexicon", *Proc. TINLAP .1.* Cambridge, Mass.

[4] D. McDonald and J. Pustejovsky (1985) "Description-Directed Natural Language Generation" *Proc. IJCAI85.* pp.799-805.

[5] Z. Harris (1968) *Mathematical Structures of Language.* Wiley-Interscience

[6] K. Kittredge and J. Lehrberger eds. (1982) *Sublanguage: Studies of Language in Restricted Semantic Domains.* deGruyter

[7] M. Chevalier et al. (1978) *TAUM-METEO.* Groupe TAUM, Université de Montréal.

What's in an Answer: A Theoretical Perspective on Deductive Question Answering*

L. K. Schubert
L. Watanabe

Department of Computing Science
The University of Alberta
Edmonton T6G 2H1

Abstract -- Questions and answers are examined with a view towards improving the quality of answers producible by a deductive question answering algorithm. Most past work assumes that question answering is reducible to theorem proving. This reduction is often accompanied by implicit assumptions about the form of the question and answer. The focus in this paper is on questions interpretable as requests for information. The form of such requests is analyzed, and desirable properties of responses defined. An answer is classified as categorial or selective, and characterized by properties of correctness, completeness, definiteness, nontriviality, comprehensibility, and informativeness. Extant deductive question answering procedures are evaluated with respect to these properties and shown to be deficient in their ability to generate certain classes of answers. The study leads to various interesting open problems, including that of devising a reasonably efficient procedure for categorial question answering.

1. Introduction

"What's in an answer?" is a question left formally unanswered by most researchers in question-answering. Usually, questions and answers are defined by informal elucidation and examples. Such definitions make it difficult to evaluate the performance of question answering systems.

Indeed, we believe that the lack of formal definitions has led to a general misapprehension about the status of deductive question answering: it is widely assumed that certain procedures for deductive question answering, to be found in any AI text that deals seriously with deduction, are logically adequate for yes-no and wh-questions (at least if the reasoning needed to prove that the answer follows from the premises can be carried out within a standard first-order framework). We will show that this procedure, due to Green, Luckham and Nilsson, is not in general adequate for wh-question answering.

One problem that arises in any discussion of question answering is how to separate question answering from question interpretation. We are not concerned here with natural language understanding, but rather with deductive information retrieval from a knowledge base. Therefore we will focus on certain kinds of formal *requests for information*, taking for granted the process by which a request might have been derived from an English utterance. The utterance might have been a question ("Who in the department knows prolog?"), an imperative sentence ("Tell me who.."), an indicative sentence ("I would like to know who..."), an exclamation ("If only you would tell me who..."), or a fragment of any of these types of sentences. Nevertheless, we will continue to refer to requests for information informally as questions, since that is their paradigmatic form in English.

* We would like to thank Jeff Pelletier and the referees for their helpful comments on the manuscript. This research was supported in part by NSERC Operating Grant A8818.

Broadly speaking, there are two types of requests for information, corresponding respectively to yes-no questions and constituent (or wh-) questions. In the first type, the question-asker is requesting the truth value of a sentence; in the second, he is requesting information about the membership of a specified set, or for multi-constituent questions, of a specified relation (as in "Who in your class loves whom?").

Evidently, any complete proof procedure is formally adequate for answering yes-no questions: we interleave a proof attempt and a disproof attempt for the sentence in question, and answer T ("Yes") if and when the proof attempt succeeds, F ("No") if and when the disproof attempt succeeds, and U ("Unknown") if the sentence is found to be logically independent of the available information, or resource constraints are exceeded before a T or F answer is determined. Let us assume that the information available for question answering is self-consistent. Then this process has the property that for any question, there exists a finite (though in general unknown) resource bound such that if the process is run with that bound or a more generous one, the answer will be T iff the sentence is a logical consequence of the available information, F iff its negation is a logical consequence of the available information, and U iff both the sentence and its negation are consistent with the available information. In other words, the question is answered as correctly and completely as the available information permits, *in the limit* as resource constraints are relaxed. Given the undecidability of first-order logic, this correctness and completeness in the limit is the best one could hope for.

In view of the formal adequacy of standard deductive methods for yes-no questions, we will focus on (single-) constituent questions.

2. Related Work

Our immediate concern is with deductive question answering procedures of the type developed by Green [Green 69] and Luckham and Nilsson [Luckham & Nilsson 71], which are well-established in AI and will be discussed in some detail in section 7.

However, questions and answers have also been extensively studied in the field of philosophical logic, beginning with Adjukiewicz's analysis of questions and answers in terms of propositional functions. Our own proposals will lean on this tradition to some extent. Like Adjukiewicz (and more recently [Bennett 79]), we take single-constituent questions to be, in essence, questions about the truth set of a given predicate. (This is defined as the *request predicate*, R, in the next section.)

Aqvist [Aqvist 65] and Hintikka [Hintikka 76] treat questions as requests for information and attempt to analyze them in terms of the knowledge states of the questioner and answerer. For example, according to Hintikka a question of form "Which x is (has property) F?" has associated with it a desired state (or *desideratum*)

$$(\exists x)K_i F(x)$$

where "K_i" means "I know that". This desired epistemic state is attained if I come to know that $F(b)$, and can truthfully assert

I know what (or who) b is.

Our own criteria for questions of this type will be somewhat weaker, requiring only that answers be as specific as the question answerer's knowledge permits, and that they avoid "private symbols" known only to the question answerer.

Belnap and Steel's concepts of category conditions, selection, completeness-claim, and distinctness-claim have analogues in our concepts of answer types, completeness, and definiteness, but require the introduction of new types of quantifiers and operators into the logic [Belnap & Steel 76].

In general, the philosophical logic literature has only been moderately useful for our endeavour, since it has little to say about the *inference* of answers to questions. A focus on deductive question answering leads to problems quite different from those considered in the philosophical literature, ones for which even an adequate formal vocabulary is lacking. Thus one of our main objectives is to begin development of such a vocabulary. As far as possible, we avoid modal notions (especially in the object language of the question asker and answerer), since our initial aim is to analyze deductive question answering based on first-order logic. However, some of our definitions make use of a computational notion of "knowing" similar to that of Konolige [Konolige 85].

Finally we should mention relevant work in computational linguistics, particularly that of Cohen and Levesque [Cohen & Levesque 85] who analyze questions as speech acts involving the goals and intentions of the questioner. We assume that the reasoning processes they describe would sometimes lead to requests for information of the type which provide the starting point for our own investigation. Moreover, goal-directed reasoning might be resumed if the initial request cannot be satisfactorily met (according to criteria such as those we shall propose).

3. The Form of Questions and Answers

The form of a single-constituent question (more accurately, request for information) expressible in first-order logic may be taken to be

$$(\text{wh } x\text{: } P(x))Q(x)$$

where P is the *presupposed category*,
Q is the *question predicate*,
$R =_{df} \lambda x[P(x) \wedge Q(x)]$ is the *request predicate*

Formally, $P(x)$, $Q(x)$ and $R(x)$ are open sentences containing x as their only free variable. The question is intuitively to be read as "Which P is a Q?" or "Which P's are Q's?". For our purposes, the wh operator is considered part of the *interaction* language but not part of the *representation* language of the question answering system. (The source of this distinction is [Levesque 84].) We will not attempt to explicate the semantics of wh directly, but only indirectly through our formalization of *answers* to wh-questions.

Our terminology is motivated by the following considerations: a question of the above form presupposes the existence of entities of the presupposed category P. For example, if we ask, "Which CMPUT 552/85 students obtained 9's?", we presuppose that there *are* CMPUT 552/85 students. If there are none, the response should be corrective, i.e., it should not simply be "None", but rather, "There are no CMPUT 552/85 students" (cf. [Wębber & Mays 83]).

Q is called the question predicate since it can be thought of as deriving from the verb phrase (i.e., the predicate) of a question expressing the request for information.

R, the request predicate which combines P and Q, turns out to be useful in the formulation of desirable constraints on answers.

Note that our formulation precludes "How...", "Why...", and "How many..." questions, among others, unless our logic treats "ways, manners, and means", reasons, and numbers as ordinary individuals. Questions such as "Which two people..." are also excluded, unless our logic treats collections (such as pairs, triples, etc., of people) as ordinary individuals. Nevertheless, the range of questions we are allowing is quite sufficient for a preliminary study of answers and their properties, and for an assessment of deductive question answering methods.

In answering a request for information of the above type, we assume that we first check whether any presuppositions of the request (including the presupposition $(\exists x)P(x)$) are violated. If we prove a presupposition to be violated, we generate a corrective response (i.e., its denial). Besides attempting to disprove presuppositions, we also attempt to prove a negative answer to the question itself, i.e., $\sim(\exists x\text{: } P(x))Q(x)$, and respond appropriately if the attempt succeeds. After this "negative phase" of the question-answering process, we attempt to provide a positive answer. (We do not discuss the negative phase further since, as in the case of yes-no questions, standard deductive methods are adequate for it; the negative phase would in practice be interleaved with the positive phase.)

The form of a positive answer may be taken to be

$$(\alpha x\text{: } A(x))Q(x)$$

where α is a quantifier and A is the *answer predicate*. (Again, $A(x)$ is an open sentence whose only free variable is x.)

If $\alpha = \forall$ (every), we have a *categorial answer*.

If $\alpha = \exists$ (some), we have a *selective answer*.

In standard first-order logic, categorial and selective answers assume the respective forms

$$(\forall x)[A(x) \rightarrow R(x)], \text{ and } (\exists x)[A(x) \wedge R(x)].$$

Categorial answers and selective answers can be thought of as corresponding respectively to plural and singular wh-constituents in the question. Thus the question "Which department members know prolog?" might prompt the categorial answer "All department members other than Jones (know prolog)", i.e.,

$$A(x) = \text{dept-member}(x) \wedge \sim(x = \text{Jones})$$

while the question "Which department member knows prolog?" might prompt the selective answer "Smith", i.e.,

$$A(x) = (x = \text{Smith})$$

or perhaps "Smith or Jones (I'm not sure which)", i.e.,

$$A(x) = (x = \text{Smith}) \vee (x = \text{Jones})$$

However, it is in general a matter of pragmatics, requiring reasoning about the goals of the questioner, whether a categorial or selective answer is desired; thus we will not further discuss this issue here.

We think that categorial and selective answers are two of the most important types (cf. [Hintikka 76]) , though not the only ones. For example, a reasonable and useful response to the first of the above questions might be "Most of the

department members (know prolog)". However, we will not consider responses involving nonstandard quantifiers like "most", which cannot be translated into ordinary first-order logic. It is worth remarking here that answers involving the negative quantifier "no", such as "No department members who know Fortran (know prolog)" need not be separately considered, since they can be regarded as categorial answers to other questions, such as "Which department members do not know prolog?". (The issue of when an answer to the *negative* form of the surface question is conversationally appropriate is again a matter of pragmatics.)

We now proceed to our characterization of answers.

4. Correctness and Completeness (or Definiteness)

Certainly correctness is an essential property of any cooperative response to a request for information.

We define an answer $(\alpha x\colon A(x))Q(x)$ to be *correct* iff

$$S_a \models (\alpha x\colon A(x))R(x),$$

where S_a is the set of facts available for question answering and "$\models$" denotes "entails" (has as logical consequence).[1] Thus, to be correct not only must the answer hold, but as well, the instances of Q provided by it must be of the presupposed category P. Note that for categorial answers and selective answers, the correctness criterion is equivalent respectively to:

$$S_a \models (\forall x)[A(x) \rightarrow R(x)]$$

$$S_a \models (\exists x)[A(x) \wedge R(x)]$$

Thus, for example, if the question is "Which countries have part of the Rocky Mountains in them?" the following answers are considered correct (where P = "country", Q = "has part of the Rocky Mountains in it", R = "country which has part of the Rocky Mountains in it"):

(a) "Canada and the US": $(\forall x\colon x = \text{Canada} \vee x = \text{US})Q(x)$

(b) "Canada": $(\exists x\colon x = \text{Canada})Q(x)$

(c) "Canada or the US": $(\exists x\colon x = \text{Canada} \vee x = \text{US})Q(x)$

(d) "Canada or Alberta":

$$(\exists x\colon x = \text{Canada} \vee x = \text{Alberta})Q(x)$$

Note that (a) is categorial, while (b) - (d) are selective. (We are taking for granted that the decision to answer categorially or selectively has been made on suitable pragmatic grounds.)

The following answers are incorrect even though they are true:

(e) "Canada or Alberta":

$$(\forall x\colon x = \text{Canada} \vee x = \text{Alberta})Q(x)$$

(f) "Alberta": $(\exists x\colon x = \text{Alberta})Q(x)$

(g) "Alberta or Ontario":

$$(\exists x\colon x = \text{Alberta} \vee x = \text{Ontario})Q(x)$$

It is worth noting that we could have imposed correctness criterion

$$S_a \models (\alpha x\colon A(x))Q(x) \text{ and } S_a \models (\forall x\colon A(x))P(x)$$

instead of the one proposed, without essential change. This requires the answer predicate to specify a subcategory of P. For categorial answers, this is equivalent to the previous criterion, but for selective answers, it is somewhat stronger: it judges (d) above to be incorrect. However, any correct answer according to the weaker criterion is easily converted to a correct answer according to the stronger criterion, by conjoining $P(x)$ to $A(x)$. For example, the following is correct according to the stronger criterion:

(d') "A country identical with Canada or Alberta":

$$(\exists x\colon P(x) \wedge (x = \text{Canada} \vee x = \text{Alberta}))Q(x)$$[2]

Another desirable property of answers is that they be as *complete* as possible. However, one has to be careful in formalizing this idea. In the case of categorial answers, it is clear that "completeness" can be identified with "maximum coverage" by A (of the positive instances of R) but in the case of *selective answers* this notion of maximum coverage is inappropriate, since selective answers *by definition* need to supply only one instance of R. What *is* desirable, however, is that this instance be as fully specified as possible. The best we can hope for is that A *uniquely identifies an instance*; in that case the existential quantifier in the selective answer can be replaced by the *definite quantifier*, "the". These considerations motivate the following definitions.

A categorial answer is *complete* iff

$$S_a \models (\forall x)[R(x) \rightarrow A(x)]$$

Note that completeness and correctness of categorial answers amounts to equivalence of the answer predicate to the request predicate, i.e.,

$$S_a \models (\forall x)[A(x) \equiv R(x)]$$

An ideal categorial question answerer should always try to prove or disprove that its answer is complete and, unless $A \equiv P$, inform the questioner as to the outcome of the proof/disproof attempt. (E.g., "Canada and the US, and no others"; or "Canada, among others"; or "Canada and possibly others".)

A selective answer is *definite* iff

$$S_a \models (\exists y)(\forall x)[A(x) \rightarrow x = y]$$

Note that whenever $A(x)$ is of form $x = c$, where c is a constant (or more generally, a variable-free term), the answer is definite (but not necessarily informative, as we shall see).

While it is clear that these properties of answers are desirable, it will also become clear, first, that they may not be obtainable in general and second, that they are insufficient to ensure that answers will be informative. We address each of these points in turn.

5. Relative Completeness (or Definiteness)

In general, the knowledge of the question answerer may not be sufficient to derive a categorial answer which is complete (and also nontrivial -- a point we take up in the next section). We therefore require a formal criterion for determining whether the answer is as complete as the available knowledge permits.

Accordingly, we say that a categorial answer is *relatively complete* iff for all formulas $B(x)$ (of the object language of the question answerer) containing x as the only free variable,

$$\text{if } S_a \models (\forall x)[B(x) \rightarrow R(x)]$$

[1] A false answer which follows from mistaken beliefs on the part of the question answerer may thus be deemed correct; but this "defect" is unavoidable in any *useful* notion of correctness.

[2] Any reader who feels that this answer is still peculiar is reminded that there can be no guarantee that the question-answerer's knowledge will include the fact that Alberta is not a country.

then $S_a \models (\forall x)[B(x) \rightarrow A(x)]$,

i.e., A is at least as general as any derivable correct answer.

Analogously, we say that a selective answer is *relatively definite* iff it is as specific as the question answerer's knowledge permits (i.e., any further constraint the question answerer is able to add is already implicit in the meaning of $A(x)$ and $R(x)$); or formally, iff for all formulas $B(x)$ containing x as the only free variable,

$$\text{if } S_a \models (\exists x)[A(x) \wedge B(x) \wedge R(x)]$$

$$\text{then } MP_a \models (\forall x)[(A(x) \wedge R(x)) \rightarrow B(x)],$$

where MP_a (a subset of S_a) is the set of *meaning postulates* available to the question answerer. (These are axioms expressing such facts as "All bachelors are unmarried", "The territory lying within the borders of any country belongs to that country", i.e., facts which hold solely in virtue of the conventional meanings of the terms they involve.)[3]

6. Nontriviality and Comprehensibility

The insufficiency of the correctness and completeness conditions for categorial answers is evident from our earlier observation that they amount to *equivalence* of R and A. Thus an answer of form

$$(\forall x: R(x))Q(x)$$

is always correct and complete, but is clearly useless (e.g., "All countries that have part of the Rocky Mountains in them have part of the Rocky Mountains in them".)

For an answer to be *nontrivial,* we would like to make sure that the questioner does not yet *know* that answer, i.e.,

$$\sim K_q(\alpha x: A(x))Q(x)$$

where we interpret "knowing a fact" as being able to verify it effectively (and presumably, quickly) by application of a "deductive recall" algorithm K to a base of facts S_q *explicitly* known to the questioner (cf. [Konolige 85]). This might be written as follows. K is a (quickly) computable function from pairs S, ϕ, where S is a set of sentences and ϕ is a sentence, to {T,F,U} (meaning "true", "false", and "unknown" respectively), where

$$K(S,\phi) = \begin{cases} T & \text{only if } S \models \phi \\ F & \text{only if } S \models \sim\phi \\ U & \text{only if neither } \phi \text{ nor } \sim\phi \text{ is in } S.[4] \end{cases}$$

In terms of K, K_q is defined as

$$K_q(\phi) \quad \text{iff} \quad K(S_q,\phi) = T.$$

The preceding criterion is still too weak, however. For example, the "Rocky Mountain" question might be answered with "Canada, and all other countries that have part of the Rocky Mountains in them." This is nontrivial if the questioner knows nothing about the Rocky Mountains, yet intuitively seems "partially trivial"; this is because there is a potentially less comprehensive answer than the one given, from which the questioner could easily infer the one given.

We therefore define a categorial answer to be *entirely* nontrivial iff there does not exist a correct categorial answer with answer predicate A' such that

$$S_a \not\models (\forall x)[A(x) \rightarrow A'(x)] \text{ and}$$

$$K_q[(\forall x: A'(x))Q(x) \rightarrow (\forall x: A(x))Q(x)]$$

This criterion is violated in the above example, with

$$A'(x) =_{df} [x = \text{Canada}],$$

$$A(x) =_{df} [x = \text{Canada}] \vee [\sim[x = \text{Canada}] \wedge Q(x)].$$

In a closely related sense, the notion of partial triviality also applies to selective answers, as in "A North or South American country with part of the Rocky Mountains in it". We therefore say that $(\exists x: A(x))Q(x)$ is an *entirely nontrivial selective answer* iff there does not exist a potentially less definite correct answer from which the questioner could easily infer the answer given; or formally, iff there does not exist a correct selective answer with answer predicate A' such that

$$S_a \not\models (\forall x)[A'(x) \rightarrow A(x)] \text{ and}$$

$$K_q[(\exists x: A'(x))Q(x) \rightarrow (\exists x: A(x))Q(x)].$$

This criterion is violated in the above example, with

$$A'(x) =_{df} [NAC(x) \vee SAC(x)],$$

$$A(x) =_{df} [NAC(x) \vee SAC(x)] \wedge Q(x),$$

where NAC and SAC mean "is a North American country" and "is a South American country" respectively.

Our definition of "Knows" involved a set S_q of axioms known to the questioner and an algorithm K which tries to determine whether a sentence lies in the logical closure of S_q. In a question-answering system, a good starting point for S_q might be the meaning postulates of the language, especially when no question context or user model is available. For this case, we can define an *analytic* answer to be one easily deducible from meaning postulates.

For example, in response to "Who is a bachelor?", the answer "An unmarried man " would be considered analytic if "A bachelor is an unmarried man " was assumed to be a meaning postulate.

However, this example also points out the fallibility of this approach: if the questioner did not know this meaning postulate and wanted to know "What is a bachelor?" then the response "An unmarried man" would be nontrivial.

Conversely, a non-analytic answer may be trivial. Consider the categorial version of the "Rocky Mountain" question. The response "All countries which maintain parks in the Rocky Mountains" is correct, non-analytic and complete but will be trivial if, for example, the questioner already knows that a country maintains parks in a mountain range only if the mountain range lies partially within its borders. (Note that this is contingent knowledge, rather than knowledge of meaning postulates.)

As Hintikka noted [Hintikka 76] it is desirable for the term designated by the answerer to be "known" to the questioner for the answer to be comprehensible. We will impose the less stringent requirement that an answer should be *comprehensible* in the sense that it contains no "private symbols". A "private symbol" may be defined as any constant symbol, function symbol, or predicate symbol of the answer formalism which has no predefined English equivalent. For a predicate logic question answering system, we can assume

---

[3] Our condition of relative definiteness is perhaps too strong. For example, if a system possesses the axioms (∃x: A(x))R(x) and (∃y: A(y) & B(y))R(y), where A and B are unrelated predicates, a selective answer based on the first of these axioms will be deemed non-relatively definite. Yet, it is in a sense the most definite answer the system can give "for the x referenced by the first axiom", since "the y referenced by the second axiom" may be an entirely distinct entity. However, weakening the criterion of relative definiteness to allow for this observation appears to be impossible without adding modal operators to the object language.

[4] This is only a partial specification of K, of course. Various other reasonable constraints could be imposed on K, such as monotonicity: if K(S,φ) ≠ U then for all ψ, K(SU{ψ},φ) ≠ U.

that all predicate symbols are "public" (non-private). However, skolem constants and functions created when translating natural language sentences into quantifier-free form will often be private.[5]

Finally, we define an *informative* answer to be one that is nontrivial and comprehensible.

We do not claim that correctness, completeness (or definiteness) and informativeness in the sense in which we have defined these properties provide either *necessary* or *sufficient* conditions for an answer to an information request to be helpful or appropriate. What we do claim is that these properties are usually *desirable* in an answer, regardless of the context in which the information is requested.

We now attempt to evaluate the theoretical adequacy of some standard deductive question answering methods with respect to our characterization of answers.

7. A Reexamination of Deductive Answer Extraction

The problem of modifying a theorem-prover to generate answers to wh-questions was considered by Cordell Green [Green 69] in his QA3 system. Green's method consisted of formulating an existential presupposition of the question in quantifier-free form, denying the presupposition, and then using a resolution theorem prover to refute the denial. An ANS predicate was then appended to the denial of the theorem to collect the composition of substitutions performed on the variables corresponding to the existentially quantified variables of the theorem. When the proof was repeated, the ANS predicate would then have substituted for its variable a term which could in some sense be considered as an answer to the original question.

Luckham and Nilsson [Luckham & Nilsson 71] proposed an extension to Green's method that produced a sentential answer, rather than just a substitution. The answer formula was either similar to the entire question formula or to one of its disjuncts (see below). They also provided a theoretical justification for both methods, showing that the answer was a logical consequence of the axioms, and that the theorem was a logical consequence of the answer.

To what extent do answers provided by the Green-Luckham-Nilsson procedure conform with the desiderata we have proposed? The first and perhaps most important point to note is that the procedure is generally unsuitable for deriving categorial answers. To take a very simple example, a system which knows nothing except the fact that all creatures are mortal should be able to supply a complete categorial answer to the question, "What creatures are mortal?". But the Green-Luckham-Nilsson procedure, which would attempt to prove that some creature is mortal, would fail to obtain an answer.

Of course, the answer predicate of any correct, definite, nontrivial selective answer also provides an answer predicate for a correct, nontrivial categorial answer. However, even in cases where both selective answers and categorial answers are entailed by the knowledge base, no *definite* selective answer may be derivable from it. Besides, a categorial answer based on a selective answer will not in general be complete or even relatively complete. Within the prolog community, there appears to be a general assumption that complete answers to questions, though not obtainable in general with any one proof, can be obtained in principle by enumerating proofs and collecting (selective) answers obtained by different proofs. However, it is clear from the preceding remarks that this assumption is mistaken. Also, a categorial answer which enumerates instances may be unnecessarily verbose. (For some ideas on compressing enumerative answers, see [Kalita et al. 84].) To the best of our knowledge, the problem of computing categorial answers is wide open at this time.

It is worth noting that a procedure is easily formulated for computing complete, informative answers which is as effective as we can hope -- though it is hopelessly inefficient. The idea is to syntactically enumerate possible comprehensible answers, and to interleave attempts to prove these possible answers and their completeness and nontriviality with the enumeration itself. (To establish nontriviality, we must of course have a model of the questioner that allows us to simulate K operating on S_q.) If a complete, informative answer is implicit in the knowledge base, this procedure will eventually find one.[6] It is an open problem whether there is also a semi-decision procedure of this type for relatively complete (and/or entirely nontrivial), comprehensible answers.

Let us now look more closely at the Green-Luckham-Nilsson procedure, and try to assess its performance with respect to selective answers. Certainly the answers are correct (in view of the theorems mentioned above), but are they relatively definite and informative?

To expect that *every* proof would yield a selective answer with these properties would be to expect too much. For example, if a knowledge base contains separate assertions to the effect that "Someone knows prolog" and "John knows prolog", either assertion can be used to prove that someone (or something) knows prolog; but the former does not produce a relatively definite, informative answer while the latter does. The best we can expect, therefore, is that *some* proof or combination of proofs will allow extraction of a relatively definite, informative answer.

The answers produced by Luckham and Nilsson's procedure are restricted to certain syntactic forms bearing a specific relation to the question posed as a theorem. If the theorem is written in quantifier-free disjunctive normal form

$$\sim C_1(X(1),G(1)) \vee \cdots \vee \sim C_n(X(n),G(n))$$

where the $X(i)$ correspond to the universally quantified variables and the $G(i)$ correspond to the skolem functions representing the existentially quantified variables in the ith clause (conjunction of literals), then the answer would be of form

$$\sim C_1(X(1),Y(1)\theta(1,r)) \vee \cdots \vee \sim C_n(X(n),Y(n)\theta(n,r))$$

where $\theta(i,r)$ denoted the composition of substitutions in the resolution proof tree on paths from the root node to the leaf node corresponding to C_i (where the clause at that leaf node is the dual of C_i obtained when the theorem is denied and converted to conjunctive normal form.)

Because of this syntactic restriction, the only types of answers that can be generated are sentences "echoing"

[5] Though they need not be. E.g., consider "There is a girl and her name is Mary". The skolem constant created in the first clause is assigned the English translation "Mary" by the second clause.

[6] This procedure is impractical in the same way that syntactic enumeration procedures for *hypothesis generation* are impractical. Indeed, categorial question answering can be viewed as a kind of hypothesis generation process: the goal is to discover a property A whose possession (nontrivially) entails a given property R. It appears that methods like Morgan's [Morgan 71] or Pietrzykowski's [Pietrzykowski 78], which treat hypothesis formation as the dual of deduction, can be adapted to our task. In the "mortal creatures" problem, for example, to determine what properties entail being mortal, we would determine what properties are *entailed by not* being mortal, and use the negation of such properties as possible categorial answers. In particular, if we predicate "not mortal" of c (a skolem constant), we immediately obtain "not a creature" for c, from the premise that all creatures are mortal, so that the negation of this property, "creature", becomes a possible categorial answer, as required. Formalization of this method may well yield an effective method for categorial question answering.

portions of the question and containing terms in the Herbrand universe satisfying the question predicate. The answer cannot be phrased in terms of other predicates, though some additional information can be derived by tracing the skolem functions back to the clauses from which they arose. This can be seen in the following example:

Example

Assume that the axioms are "All creatures are mortal", "There is a creature", and the question is "Who (or what) is mortal?".

To answer this question, let $M(x)$ denote that x is mortal, and $C(x)$ denote that x is a creature. The denial of the presupposition of the question is "Nothing is mortal". Then the axioms can be written in clause form as

$$\sim C(x) \vee M(x),\ C(c), \text{ and } \sim M(x)$$

where c is a skolem constant. The Luckham-Nilsson method would then generate the answer $M(c)$ which means "Something is mortal". This can be viewed as a selective answer, with answer predicate $\lambda x[x=c]$. The answer is correct and definite but, if c is not a public constant, is not comprehensible. Here a "deskolemization" procedure due to Cox and Pietrzykowski [Cox & Pietrzykowski 84] could be used to eliminate c in favour of an existentially quantified variable, but then the result will no longer be definite. (In fact, in this instance there can be no definite, comprehensible answer, since no uniquely identifying characteristics for c are available.)

Green was aware of this problem and attempted to repair the difficulty by supplementing the "formal" answer with additional information. For the previous example, the answer predicate $ANS(c)$ would be supplemented by the clause in which the constant c was created, $C(c)$, or perhaps the English language version of the clause, "There is a creature."

Another example from [Chang & Lee 73] further illustrates the sorts of answers obtained by the Green-Luckham-Nilsson method, and the problem of ensuring that answers will be comprehensible.

Example

Assume that the answerer knows:

"Everyone who entered this country and was not a VIP was searched by a customs official."
"William was a drug pusher."
"William entered this country."
"William was searched by drug pushers only."
"No drug pusher was a VIP."

Suppose the answerer is asked "Who was both a drug pusher and a customs official?" To answer this question, let $S(x,y)$ denote x was searched by y, $V(x)$ denote x is a VIP, $E(x)$ denote x entered the country, $C(x)$ denote x is a customs official, and a denote William. The answer produced by the Luckham-Nilsson method would be

$$D(f(a)) \wedge C(f(a))$$

which means "$f(a)$ is a drug pusher and a customs official" Unfortunately, the questioner might have no idea who is $f(a)$.

If Green's method is used, the answer $ANS(f(a))$ would be generated. In response to a request for further information, the answerer might provide "Everyone who entered this country and was not a VIP was searched by a customs official." This is insufficient to determine a relatively definite, comprehensible answer. The answer we desire is

$$(\exists x\colon S(a,x) \wedge \sim V(x))D(x) \wedge C(x),$$

i.e., someone who searched William and is not a VIP is a drug pusher and a customs official (where we have assumed that none of the axioms are meaning postulates). Thus the clauses that ought to have been added to the answer extracted from the proof are $S(a,f(a))$ and $\sim V(f(a))$; from these the formula $A(x)$ $(=S(a,x) \wedge \sim V(x))$ needed for a relatively definite answer could have been derived, with the aid of a de-skolemization procedure. (A more general algorithm would be required than that in [Cox & Pietrzykowski 84] which processes individual literals only.)

Does Green's method of supplementing answers at least guarantee that *some* proof of any given question will yield a relatively definite, comprehensible answer? Unfortunately, the answer is negative, as the following very simple counterexample shows. Suppose that the question answerer possesses just two facts, namely, that Agatha was murdered by someone and that only a person who drives a dumptruck could have murdered Agatha:

$$M(c,a),\ \sim M(x,a) \vee D(x)$$

where c is a skolem constant. Then the *only* proof that answers the question "Who (or what) murdered Agatha?" is one that resolves the denial clause $\sim M(x,a)$ against the first of the two axioms. The fact that is needed, however, to flesh out the answer $M(c,a)$ to one that is relatively definite is $D(c)$ (or, more redundantly, $\sim M(c,a) \vee D(c)$). This fact is not a substitution instance of any clause in any proof, or of a base clause involving a private symbol that occurs in the answer.

8. Conclusion

We have introduced a set of formal concepts for assessing the quality of answers returned by deductive question answering procedures. In particular, we have formalized some desirable properties of two natural classes of answers, which we termed "categorial" and "selective". We have shown that standard question answering methods are in general unable to produce categorial answers, and that they also fall short in the kinds of selective answers they produce.

We have mentioned some open problems, the most important of which is the formulation of reasonably efficient procedures for categorial question answering. In general, our work indicates that there are unsuspected depths, and many unexplored issues, in the area of deductive question answering.

References

Aqvist 65.
Lennart Aqvist, *A new approach to the logical theory of interrogatives*, University of Uppsala, 1965.

Belnap & Steel 76.
Nuel D. Belnap and Thomas B. Steel, *The logic of questions and answers*, Yale University Press, New Haven, 1976.

Bennett 79.
Michael Bennett, *Questions in Montague Grammar*, Indiana University Linguistics Club, Bloomington, Indiana, 1979.

Chang & Lee 73.
Chin-liang Chang and Richard Char-tung Lee, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.

Cohen & Levesque 85.
Philip R. Cohen and Hector J. Levesque, Speech acts and rationality, *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, 1985,

49-60.

Cox & Pietrzykowski 84.
P. T. Cox and T. Pietrzykowski, A complete, nonredundant algorithm for reversed skolemization, *Journal of Theoretical Computer Science 28*, (1984), 239-261, North-Holland.

Green 69.
Cordell Green, Theorem-proving by resolution as a basis for question-answering systems, in *Machine Intelligence*, vol. 4, Bernard Meltzer, Donald Michie and Michael Swann (ed.), Elsevier, New York, 1969, 183-205.

Hintikka 76.
Jaakko Hintikka, *The Semantics of Questions and the Questions of Semantics*, North-Holland, Amsterdam, 1976.

Kalita et al. 84.
J. K. Kalita, M. J. Colbourn and G. I. McCalla, A response to the need for summary responses, *Proceedings of the 22nd Annual Meeting of the Association for Computational Linguistics*, 1984, 432-436.

Konolige 85.
Kurt Konolige, A computational theory of belief introspection, *IJCAI-85 1*, (1985), 502-508.

Levesque 84.
Hector J. Levesque, Foundations of a functional approach to knowledge representation, *Artificial Intelligence 23*, (1984), 155-212.

Luckham & Nilsson 71.
David Luckham and Nils J. Nilsson, Extracting information from resolution proof trees, *Artificial Intelligence 2*, (1971), 27-54.

Morgan 71.
Charles G. Morgan, Hypothesis generation by machine, *Artificial Intelligence 2*, (1971), 179-187.

Pietrzykowski 78.
T. Pietrzykowski, Mechanical hypothesis formation, Res. Rept. CS-78-33, Department of Computer Science, University of Waterloo, Canada, 1978.

Webber & Mays 83.
Bonnie Lynn Webber and Eric Mays, Varieties of user misconceptions: detection and correction, *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, 1983, 650-652.

A New Implementation for Generalized Phrase Structure Grammar

Philip Harrison and Michael Maxwell

Boeing Artificial Intelligence Center
Boeing Computer Services, M/S 7A-03
PO Box 24346
Seattle, WA 98124

Abstract

We describe the syntactic component of a large natural language processing system currently being implemented at the Boeing Artificial Intelligence Center. The system is based on the latest version of Generalized Phrase Structure Grammar, but includes some modifications. The system has features that enable grammars to be easily developed. It also has features that promote parsing efficiency, efficient handling of certain syntactic ambiguities, and the maintenance of a large lexicon.

1. Introduction

In this paper, we describe a linguistically based natural language processing system that has been under development at the Boeing Artificial Intelligence Center since the summer of 1984. Our methodology and goals are similar to other work that has recently been reported in the literature, e.g., Rosenschein and Shieber [10] , Schubert and Pelletier [11], Gawron et al [3], Proudian and Pollard [8], and Pulman [9]. These efforts all rely on current nontransformational syntactic theory as a basis for parsing, and some version of formal logic as the basis for the translation of parser output into meaning representations. We all share the goals of producing modular programs for handling syntax, semantics, and pragmatics and of encoding rules of grammar and other specialized information in external, easily comprehensible data files so that the system is easy to modify and expand. We also have the goal of system transportability with respect to operating environments, domains of discourse, and various functional applications such as database question-answering or interfaces to complex software systems.

Our own work is based on a variant of Generalized Phrase Structure Grammar (GPSG) as outlined in Gazdar, Klein, Pullum and Sag [4]. (This book will hereafter be referred to as GKPS). The current implementation consists of a bottom-up parser written in Franz Lisp, a grammar with approximately two hundred rules, and a rudimentary semantic interpeter that generates logical formulas along the lines advocated by Schubert and Pelletier. There is at present no pragmatic or discourse processing. All of the components have been embedded in a database question-answering system. There are several aspects of our approach that are unique: the formulation of the underlying syntactic system, the parsing algorithm, and the algorithm for generating database queries from logical form. We have not yet expended much effort on the semantics component, so this report will focus on syntax.

The GPSG approach to syntax can be characterized as a methodology in which large context-free (CF) grammars for natural languages are defined through a high-level formalism that provides a compact description of both the set of CF productions and the nonterminal vocabulary. (We shall use the term *production* to refer to a rule from a CF grammar.) The high-level formalism includes both a method for writing *rule schemata* and a set of *instantiation rules* that govern the determination of fully specified nonterminals from partial descriptions given in the schemata.

It is possible, in principle, to compute the actual CF grammar corresponding to a set of rule schemata and instantiation rules, although the resulting CF grammar might contain a very large nonterminal vocabulary and a very large set of productions. It is conceivable that for some GPSGs it may not even be possible to estimate the size of the underlying CF grammar without actually generating it. Fortunately, it is not necessary to have the CF grammar "in hand" in order to implement a parser; the rule schemata and instantiation rules can be interpreted as prescriptions for building phrase structure trees and the required productions of the CF grammar can be computed dynamically as tree construction proceeds. Such parsers can be efficient in practice, even though theoretical complexity analyses of worst case situations might be very discouraging.

To define a GPSG, three things must be done:

1) The nonterminal vocabulary must be defined. In introductory textbooks the nonterminals are *atomic symbols*. However, there is nothing in the definition of a CF grammar that requires the nonterminals to be atomic symbols. In GKPS the nonterminals are sets of ordered pairs of the form *<feature-name, feature-value>*. These *feature sets* are constrained to be partial functions and there are additional constraints called *feature co-occurrence restrictions*. Even more complicated definitions of the nonterminal vocabulary are in use by other investigators. Shieber (1985) discusses systems where the nonterminals are directed acyclic graph structures. In general, the nonterminals can be complex objects with various properties, slots, values, etc. We have adopted a definition of the nonterminals that imposes more structure than the GKPS definition but less structure than the directed acyclic graphs.

2) The methodology for writing schemata needs to be specified. The schemata resemble phrase structure rules, but they also allow various formal devices that are interpreted as constraints that nonterminals must satisfy in order to be substituted into the schemata. In a CF production, fully specified nonterminals must appear, so the move that is made in GPSG is to have the schemata *partially* specify the nonterminals. In most cases there are several nonterminals that can be substituted for each term of a schema in order to produce CF productions, so each schema generates a set of CF productions. In GKPS, the schemata are in what is called "Immediate Dominance, Linear Precedence" (ID/LP) format: the specification of the order of constituents is given separately from the description of their syntactic properties. Also, there is a system of *meta-rules* that takes an initial set of schemata and generates additional schemata. In our system we encode subnode order directly in the schemata and we do not use meta-rules.

3) The rules of instantiation must be given. Generally they are expressed informally and are hard-coded into the parser. They provide an interpretation of the constraints that are imposed declaratively by the schemata. Instantiation rules can also be given that impose constraints that are not represented directly in the schema formalism. An example of such a rule in GKPS is "the head constituent passes its head features up to the parent node." Instantiation rules can be quite elaborate, making reference to various properties and attributes that constituents in a parse tree must have.

The *implementation* of a parsing system involves two additional tasks:

4) A parsing algorithm must be chosen. Although several good algorithms for CF grammars have come out of theoretical computer science, there is still much that can be done to exploit special properties of natural languages.

5) The algorithm must be implemented, along with utilities for managing the schemata and the lexicon; a significant set of schemata and a lexicon must be written. This last step requires more effort than any of the others.

Much of the research work that involves the development of a large natural language parsing system can be viewed as an exploration in the enormous space of possible high-level grammars and implementations. Each implementation involves variations in all of the steps above.

In addition, it is possible to develop systems that recognize non-CF languages. The evaluation of these efforts is not easy because each of the implementation steps and the final system performance depend on how well all of the prior steps have been carried out in the context of an intended end-use application. Some questions that are useful in making an evaluation include

1) Is the formalism easy to learn, understand, and use?

2) Can the system parse a reasonable natural language subset?

3) Are there inherent intractabilities in the proposed scheme?

2. A Description of the Vocabulary for the Induced CF Grammars

The terminal vocabulary of our system consists of printed words of English, plus punctuation signs. We have an input processing program that reads the sequence of characters typed by users in conventional format and transforms the input into a list of Lisp atoms by separating punctuation from the preceding words.

We take the nonterminal vocabulary to be 4-tuples of the form

<pos, features, gap, foot-features>

where each position will be referred to as a *property* of the nonterminal, i.e., the first position will be referred to as the *pos* property, and analogously for the remaining positions.

The possible values for the *pos* property are taken from a set D_{pos} of *atomic symbols*, each of which designates a "part of speech." These are symbols like N, V, P, A, NP, PP, etc., that linguists standardly use when referring to syntactic categories.

The *features* property must have as its value a *feature set*. A *feature* is defined to be either an atomic symbol (an *atomic feature*) drawn from a set D_a or an ordered pair of the form *<feature-name, feature-value>* (a *compound feature*). The feature names are drawn from a set of atomic symbols D_n. For each feature name γ, there is an associated set D_γ of atomic symbols from which possible values may be selected. A *feature set* is a subset of $D_a \cup \{ <\gamma, \beta> \mid \gamma \in D_n, \beta \in D_\gamma\}$ subject to the constraint that the set of ordered pairs is a partial function. That is, if $<\alpha, \beta>$ and $<\alpha, \gamma>$ are in a feature set, then $\beta = \gamma$. The sets D_a, D_n, and D_γ for each $\gamma \in D_n$ are determined by the *use* of the symbols in the schemata and by their use in the *lexical entries* for the words of the terminal vocabulary.

A feature set is then a collection of symbols and/or ordered pairs. In our Lisp-based parsing system we represent feature sets as flat lists (*feature lists*) having the form

($symbol_1$ $symbol_2$ ... $symbol_n$)

where if $symbol_i$ is a feature name then $symbol_{i+1}$ is its value. For example, suppose a feature set consists of the following set of two ordered pairs and an atomic feature:

{ <^PLU, NEG>, <^PER, 3>, REFL}.

(For convenience, we begin feature names with the character '^'.) One allowed representation of this feature set would then be

(^PLU NEG ^PER 3 REFL).

The ordered pair information has been encoded in the ordering of the list and is recoverable through the use of the list of feature names. By convention, the atom 'nil' will denote the empty set. The computation of feature sets occurs in a well-defined and efficient manner that takes feature sets stored in the lexicon and proceeds to higher nodes according to instantiation rules outlined below.

The value of the *gap* property of a nonterminal can be either the atom 'nil' or a special 4-tuple of the form $<\alpha, \beta$, nil, nil> where α = NP, PP, or AP and β is a feature set. (The restriction to NP, PP, or AP is one that can be easily modified.) The determination of the values for α and β is done according to rules of instantiation outlined below. A non-nil *gap* property for a nonterminal is our way of representing the *slash categories* of GPSG and indicates that the subtree dominated by the nonterminal contains an *unbound gap*. We allow nonterminals of the form $<\alpha, \beta$, nil, nil> to dominate the empty word Λ, and when such nonterminals (*gap nodes*) are inserted in a parse tree, they become the value of the *gap* property of the parent and higher nodes until the gap is *bound* by a schema which *specifies* a non-nil gap property. The precise mechanism is defined below.

The problem of knowing when to postulate a gap node during parsing is significant. If gap nodes are postulated too freely, a parser can become inefficient. Generally, it is possible to rely on the presence of particular structures that precede and signal a gap to the right.

The value of the *foot-features* property is also a feature set, but the number of possible features that are actually employed is very small, and most nodes in a typical tree have 'nil' for the value of this property. The property is used with only a few syntactic contructions, such as extraposition. An example is provided in section 3.

3. An Overview of the Schemata and the Parser

Our parser constructs trees by creating *phrasal nodes*, which contain all the properties of a nonterminal but in addition include information that is of crucial use to the parser: which input words are covered, what the subnodes are, which elements of the grammar rule remain to be matched, and other items.

There are several types of nodes that will be distinguished, in addition to the previously mentioned *gap nodes*. The two principal kinds are the *lexical nodes* and the *rule nodes*. Lexical nodes are created when the words of the input are scanned. The lexical entry for each word includes a list of nonterminals that can dominate that word in a parse tree, so whenever a word is scanned, a node is created for each nonterminal in the lexical entry. There are no rule schemata associated with these nodes. A rule node is one that has a grammar rule schema associated with it and has subnodes that are assigned because they satisfy (or *match*) the conditions specified by the schema. We also distinguish two kinds of rule node: a *partial node*, which does not have the full complement of subnodes as specified by the associated schema; and a *complete node*, which *does* have all of its subnodes.

The grammar rule schemata themselves are specified by Lisp expressions of the form

(rule <rule name> <parent specification> -->
<sequence of child specifications>)

In the following discussion, we will use the word *rule* to refer to a schema, if it is clear from the context that there can be no confusion with *rule of instantiation*.

In each rule, <rule name> is a unique atomic symbol. A *parent specification* is a list that begins with a part of speech (or the reserved symbol 'x' which is used in the rules for coordinate structures and will not be discussed here). Each specification list can also optionally contain a series of *keywords* and *values*. A *child specification* (which we shall also call a *term*) can be either a word of the terminal vocabulary (an *explicit term*) or a list beginning with a part of speech and optionally containing some keywords and values (a *phrasal term*). An explicit term can be matched by an occurrence of the word in the appropriate place in the input. In the case of a phrasal term, a match can occur with a phrasal node that satisfies constraints that the term specifies. The parts of speech that begin the descriptions specify the *pos* property of the nodes. A simple example is

(rule s1 (s) --> (np)(aux)(vp))

which says that a node with part of speech 's' can be constructed having three subnodes with parts of speech 'np', 'aux', and 'vp'. The unspecified properties (*features, gap* and *foot-features*) are free to take values within the constraints imposed by the rules of instantiation, so there are many parse trees that can be built from this rule. The keywords (which can appear in arbitrary order) and their associated values provide information about the order of subnode matching, feature set construction, subnode acceptability, and gap construction.

In our implementation, we treat the rule schemata as abstract objects consisting of properties with associated values. Two of the properties are central in controlling the matching process: the *trigger index* and the *match direction*.

The goal of the parser is to find sequences of adjacent phrasal nodes and/or words that match the sequence of terms on the right-hand side of grammar rules. (Two nodes are adjacent if they cover contiguous segments of the input sentence.) The fundamental problem in constructing parse trees is to decide which rules to check, and after selection, to decide the order in which to try to match the terms. Both of these issues are resolved through a system of *indexed rule triggering* and *ordered matching*.

Rule triggering is the act of selecting a rule in order to check whether it actually describes the grammatical structure of the input sentence. The rule property that governs triggering is the *trigger index*, whose value is a number that specifies which term on the right-hand side is the first to be matched. The trigger index is given in the parent specification by including the keyword *trigger-index* followed by the number of the trigger term. If a rule is *unsubcategorized* (which is the default), then its trigger term is stored in a data structure called the *trigger matrix*. Currently, rule triggering is done whenever the parser scans a new word, creates a lexical node, or creates a complete rule node. We distinguish two kinds of triggering: *general* (or *unsubcategorized*) triggering and *lexical* (or *subcategorized*) triggering. In the case of general triggering, the triggering item (word, lexical node, or complete node) is checked against all of the trigger terms stored the trigger matrix. Whenever the triggering item matches a trigger term, the rule is retrieved from the matrix and the parser creates a new parent node with the triggering item as its only subnode. At present there are no heuristics to guide the triggering process by selecting only the most promising rules and deferring the others. It should be possible to speed up the parser significantly by including such heuristics. We have not done so yet because the parser response time has been adequate for our development purposes. (Parse times for our test sentences are under a second on both the Sun Workstation and the Vax 11/780.)

Lexical triggering can occur only with lexical nodes. Whenever a phrasal node is created from a lexical item, any rules listed under the 'rules' property of the lexical item are also triggered. In this case we will say that the lexical item has triggered the rules. Rules that are subject to lexical triggering are declared through the use of the keyword *subcategorized* in the parent specification and are not stored in the trigger matrix. Lexical triggering is used with verbs and adjectives that subcategorize their complements.

In general, our parser does not assign the subnodes in left-to-right order and in this respect is similar to the "head first" parser reported by Proudian and Pollard [9]. However, we allow rule triggering to be specified for any term, although care must be exercised to not trigger a rule on a constituent that could potentially be matched by a gap node. For a newly triggered rule, if the trigger term is not the first term on the right-hand side, then the parser will eventually have to look for nodes to the left of the trigger node in order to match those terms. Since the parser processes sentences from left to right, potential subnodes will have already been constructed.

After a rule has been triggered and the trigger term has been matched, the parser uses the value of the rule's *match direction* to determine the order in which any remaining terms will be matched. This rule property can be set by using the keyword *match-direction* followed by 'L' or 'R' in the parent specification. A value of 'R' tells the parser to first match all terms to the right of the trigger term, while a value of 'L' means first match to the left. The matching can be accomplished by finding appropriate words and phrases that cover additional words of the input sentence, or by creating a *gap node*, which covers no input word. For each new term that is successfully matched, the parser creates a new phrasal node that has all of the matching nodes as subnodes. The heuristics that determine when gaps will be postulated are rather complex and will not be discussed here.

The trigger index and the match direction provide two parameters for each rule that can be optimized to improve the parser's performance. In general, a rule should be triggered on the term whose presence in the parse gives the highest probability that the rule actually applies. In this way it is possible to minimize the number of times a rule is triggered without subsequent matching of all its terms. The optimization process can best be done by conducting parsing experiments with a representative corpus of textual data, although in some cases it is obvious that a rule should be triggered on a certain element (e.g. punctuation, or a coordinator).

3.1. Discussion of the Parsing Algorithm

The parsing algorithm is similar to the algorithm used in Chart parsing, except that the data structures for storing intermediate results are somewhat more complicated than a basic Chart. These data structures are required for handling the two kinds of nodes that were previously defined: *partial nodes*, and *complete nodes*. Whenever the parser creates a new partial node (say by matching the trigger term of a rule) and the next term to be matched is to the right, then the partial node is stored in a cell of the *expectation matrix*, whose rows and columns are indexed according to the part of speech of the next term and the word number of the first word in the input that needs to be covered by the next subnode. The expectation matrix is necessary because in processing sentences from left to right, potential candidates to match the next term will not yet have been created.

Once a partial node has been created, it is never modified. Whenever the parser creates a new complete node or a new lexical node, it checks the expectation matrix to see if the new node matches the next term for any partial nodes. If it does, new parent nodes are created that include the new subnode. In this way multiple parses can be created for ambiguous sentences.

When matching needs to be done to the left, as is the case whenever a rule is triggered on a term other than its first, the parser consults the *phrase matrix* to look for matches. The phrase matrix contains complete nodes in cells according to their part of speech and the number of the *last* input word they cover. All of the possible candidates that can match terms to the left are found by checking the appropriate cell of the phrase matrix.

For purposes of efficiency, it is desirable that alternative parses be weeded out as early as possible. In the presence of significant ambiguity, an all-paths parser such as this one can experience a significant degradation in response time. Unfortunately, natural discourse is full of instances where relative clauses, prepositional phrases, compound nouns, etc have a multiplicity of possible attachments that can be resolved only by appealling to semantics. Two solutions for speeding up the parser are: (1) create a semantic disambiguator that is called by the parser when an attachment choice needs to be made, or (2) constrain the parser so that only one "canonical" parse is returned. In the second instance, the parser could follow a policy of right-most attachment and allow the semantic component to reattach nodes after the parser is finished. We believe that both of these strategies will be necessary in advanced natural language processing systems. In particular, relative clause attachment is probably best done "on the fly." On the other hand, selecting the best structural description for compound nouns is best done after the parser has constructed a right-branching tree for the compound.

While we do not yet have a semantic disambiguator, right-most attachment can be specified as a side effect of our system of keywords and rules of instantiation.

3.2. Rules of Instantiation and the Keywords

Most of the instantiation rules that govern the construction of nodes are associated with specific keywords. However, two major instantiation rules are not reflected in the keywords. The first is the *foot feature passing rule*: the *foot-features* of a subnode are passed to the *foot-features* of the parent unless the part of speech of the parent is S, AP, or VP, in which case the *foot-feature* set of the child is appended to the *features* property of the parent. The second instantiation rule is the *gap percolation rule*: a node is assigned a gap node as the value of its *gap* property in two possible cases: a) it immediately dominates the gap node, or b) the node dominates a subnode that has a non-nil *gap* property and the subnode's *gap* property was not *required* by the grammar rule, in which case the parent obtains its *gap* property from the subnode. Furthermore, a tree is not acceptable as a parse of the entire input sentence if its *gap* property is non-nil. This is the standard GPSG method for handling unbounded dependencies. There are other minor instantiation rules which we will not discuss here.

3.3. Keywords Appearing in the Parent Specification

There are five optional keywords that can appear in a parent description: *features, f-features, trigger-index, match-direction*, and *subcategorized*. Only the keyword *subcategorized* is not followed by a value. We have already discussed the keywords *trigger-index, match-direction* and *subcategorized*.

The value that follows either the keyword *features* or the keyword *f-features* must be a list representing a feature set. The list is appended

to the *features* property of the parent node if the keyword is *features*; if the the keyword is *f-features*, the list that follows is appended to the *foot-features* property of the parent. This mechanism allows rules to introduce feature sets that have not been passed up from the children.

An example of the use of *features* is the following rule:

(1) (rule VPto (VP features (to)) --> to (VP))

This allows us to distinguish infinitival VPs of the type "(John tried) *to go*" from other VPs by the presence of the feature *to*. (Note that we assume the analysis first proposed by Bresnan [1] in which subjectless infinitivals are treated as VPs, rather than Ss with a missing subject (the latter as advocated by e.g. Koster and May [8]).)

An example of the use of *f-features*, in conjunction with keywords appearing in child specifications, will be given in the next section.

The following rule exemplifies the use of the keywords *match-direction* and *trigger-index*:

(2) (rule S6 (S trigger-index 2 match-direction R) -->
(S) |,| (SAdvP))

Postposed sentential adverbial phrases (SAdvP) are always set off in speech by comma intonation; we assume here that this will be reflected in writing by the presence of a comma. (In practice, this would probably not always be the case.) If it were not for the trigger-index of two, this rule would be triggered every time an S node was parsed. The trigger-index of two, corresponding to the position of the comma, means that this rule will be triggered only when a comma is found. The match-direction of R[ight] means that the rule will first attempt to match an SAdvP to the right. Presumably SAdvPs are less common than sentences, and hence more diagnostic of this construction than looking to the left for an S. That is, rule S6 can be eliminated from consideration more easily by looking to the right for a sentential adverb than by looking to the left for an entire S.

The use of the keyword *subcategorized* was described earlier. We illustrate its use here with a simple rule for adjectives; this rule is used to parse adjective phrases like "tough to talk to" and "pretty to look at".

(3) (rule A^5 (A^ subcategorized) -->
(A) (VP efeatures (to) egap (NP)))

The keywords *efeatures* and *egap* are described in the next section; their effect in this rule is to say that the VP complement must be a to-VP and must contain an NP gap. The keyword *subcategorized* indicates that this rule should be triggered only by those adjectives (the "A" on the right-hand side of the rule) that list rule A^5 in their lexical entries.

3.4. Keywords Appearing in the Child Specifications

The optional keywords that can appear in a child specification include: *passfeatures, nogap, efeatures, xfeatures, agree*, and *egap*. Neither *passfeatures* nor *nogap* is followed by a value.

The keyword *passfeatures* indicates that the value of the *features* property of the subnode must be appended to the value of the *features* property of the parent node.

For instance, the features of the verbal head of a VP are percolated up to the parent VP node by the following rule:

(4) (rule V1 (VP subcategorized) --> (V passfeatures) (NP))

Thus, if the verb "likes" bears the features "3s" (for third person singular subject), the VP which it heads will also bear that features, by rule V1.

The keyword *nogap* has two possible interpretations, depending on the value of the part of speech given in the term: if the part of speech is 'S' then the node cannot contain a left corner gap (this provides for the so-called "*that*-trace filter" of English); if the part of speech is any other value, then the node cannot be a gap node, nor can it contain an unbound gap.

Consider the following rule illustrating the use of *nogap*:

(5) (rule V2 (VP subcategorized) -->
(V passfeatures) (NP nogap) (NP))

In this context, the meaning is that the NP can neither be a gap nor contain a gap. This corresponds to the well-known opacity with regard to extraction of the first NP following verbs like "give" and "show." That is, given the following sentence--

(6) What did John give his computer?

--the only possible interpretation is that John gave something to his computer, not that John gave his computer to something, despite the fact that both "John gave his computer something" and "John gave something his computer" are possible sentences.

With regard to the use of *nogap* with S nodes, consider the following rule for S-bar (which we write "S^", for typographical reasons):

(7) (rule S^1 (S^) --> that (S nogap))

A comment on our linguistic approach is in order here. Rather than having a complementizer node which can be filled by that or a *wh*-phrase, we specify that as the unique morpheme which can precede the S constituent in rule S^1. *Wh*-phrases, in contrast, are generated by a different rule, and in fact together with their S sisters form a different category. We thus depart from the view (first proposed in Bresnan [2]) that non-*wh*-complementizers like that and *wh*-complementizers like who are dominated at the level of surface structure by the same node (COMP). We are not the only ones to make this move; a somewhat similar distinction is proposed in Chomsky [3].

Returning to the rule given in (7), observe that the S is marked with *nogap*. We will therefore not parse the following sentence as if it contained a subject gap--

(8) Who do you think that left?

--i.e. there is no parse of (8) meaning "Who do you think left?" The only possible parse of (8) is with "left" interpreted as a transitive verb having a direct object gap, and with "that" as the subject of "leave." This is to meant to capture the so-called "that-trace" filter (more properly, the complementizer-trace filter, or the fixed-subject property of Bresnan [1]).

The keyword *efeatures* must be followed by a list of atoms that are atomic features or feature names. The list does not represent a feature set; rather, the requirement is simply that the atoms in the list appear in the feature set stored under the *features* property of the node matching the term. Thus this keyword is used to require the presence of certain atomic features, or certain compound features, without specifying the feature values for the compound features.

As an example of the use of *efeatures*, consider the following rule:

(9) (rule VP10 (VP subcategorized) -->
(V passfeatures) (VP efeatures (to)))

This rule would be subcategorized by verbs like *try, want, hope*, etc. The *efeatures (to)* on the VP complement ensures that the VP in question is a to-VP, not a VP which is tensed, gerundive, etc.

The keyword *xfeatures* must be followed by a list of the same kind as follows *efeatures*. The interpretation is that the listed atomic features or feature names are *forbidden* to be in the *features* property of the subnode.

For example:

(10) (rule NP5 (NP) --> (NP) (VP xfeatures (^tense)))

This rule attaches gerundive, passive, and infinitival VP complements to NPs, as in "the man running up the hill," "the man loved by his friends," and "the man to fix the plumbing." The *xfeatures (^tense)* blocks this rule if the VP is tensed, thus preventing us from parsing "the man ran up the hill" as an NP.

Next we give an example of the use of the keyword *f-features* (which appears in the parent specification) along with the keywords *efeatures* and *xfeatures*. The keyword *f-features* is used primarily for determining the appropriate attachment of extraposed modifiers. Consider the problem of parsing the following sentences containing extraposed relative clauses:

(11) a. A man came in whom we both recognized.

b. John saw a man yesterday whom we both recognized.

In our analysis, relative clauses may be extraposed only from NPs containing an N-bar level. (This includes both NPs containing common nouns, as in "the man" or "dirt," and NPs containing words like "someone" or "everyone.") The crucial rules for parsing sentences like those in (11) are the following, in somewhat simplified form:

(12) a. (rule NP6 (NP f-features (rc)) -->
(Det passfeatures) (N^ passfeatures))

b. (rule S8 (S) --> (S passfeatures efeatures (rc)) (RelClause))

c. (rule VP5 (VP) -->
(VP passfeatures xfeatures (to) efeatures (rc))
(RelClause))

Rule NP6 states that when a determiner (Det) is attached to an N-bar (N^), the resulting NP has the foot feature "rc," which is mnemonic for "relative clause." When this NP is embedded in a VP or an S, the foot feature on the NP is automatically turned into a head feature on the VP or S. The rules for attachment of extraposed relative clauses, rules S8 and VP5, allow an extraposed relative clause to attach to an S or VP only if that S or VP has the (head) feature "rc." (Similar rules govern the attachment of extraposed PPs.) Thus, such attachment will only be possible if the VP or S contains an NP with the foot feature "rc." Since rule NP6 (and a similar rule for determinerless NPs containing common nouns or words like "someone") is the only rule introducing the foot feature "rc," the net result is that a parse containing an extraposed phrase will only be constructed if a candidate NP exists in the VP or S from which the phrase in question might have been extraposed.

Extraposition from subject NPs is also blocked if the VP contains an NP (Gueron [6]); foot features may also be used to implement this constraint as well, although doing so is somewhat more difficult.

Other constructions which may be treated by means of foot features include extraposed result clauses, as in "So many people came that we ran out of popcorn" and "Enough people came to fill up a bandwagon," and similar constructions containing "as...as", "too...to VP", "more...than" etc. (cf. Gueron and May [7]).

The keyword *agree* must be followed by a list of *feature names* that specify a set of compound *agreement features*. If a feature name is actually present in the *features* property of the subnode, then the compound feature is collected in a special slot of the parent along with the other agreement features that might be stipulated and present on the other subnodes. If the specified agreement feature is not present in the subnode, nothing is sent to the parent. The resulting collection of ordered pairs at the parent must be a partial function for the parent to be acceptable. That is, values of the agreement features from the subnodes must not differ for the same feature name. If one of the subnodes is a *gap node*, then the *features* property of the gap node is determined by the values of the agreement features that occur on the other subnodes. This allows gap nodes to carry agreement information up the tree. Finally, any agreement features that reach the parent are appended to the *features* property of the parent node. This allows agreement to be enforced in constructions with extraposed relative clauses.

A simple example of the use of the keyword *agree* is the following rule:

(13) (rule S0 (S) --> (NP agree (^number)) (VP agree (^number)))

This rule enforces number agreement between the subject and the VP. Additionally, the S node is assigned the value of ^number from the subject and/or the VP.

The keyword *egap* must be followed by a list beginning with a part of speech followed optionally by the keyword *agree* and a list of agreement features. This construction is used to constrain the *gap* property of the candidate to be a gap node with the specified part of speech. Also, if the agreement features are present on the gap node, then their values must equal the values of the agreement features stored on the other subnodes. The use of *egap* blocks the percolation of the gap node to the parent's *gap* property. An example is given by a possible rule for relative clauses:

(14) (rule rel1 (NP) -->
(NP agree (^number)) (S egap (NP agree (^number))))

This rule says that a parent node with part of speech NP can dominate two child nodes. The first child must also have part of speech NP and the second must have part of speech S. In addition, the second child must have a gap node as the value of its *gap* property and the part of speech of the gap node must be NP. Finally, if the gap node has a compound feature with feature name 'number' stored under its *features* property and if the first child also has that compound feature, then the two values must be equal. This rule would allow parsing relative clauses with null COMP nodes, such as "(the man) *you saw*." This completes the consideration of rules; we now turn to the use of lexical entries in the grammar.

4. Examples of Lexical Entries

4.1. Simple Cases

Consider first the following simple lexical entries:

(15) (word fast ((A rules (A^4 A^5) er faster est fastest)
(VAdv er faster est fastest))
(word will ((Aux features (^tense present aux modal)))
(N plural wills))

The lexical entry for "fast" implies that it can be an adjective (A), with an -er (comparative) form "faster," and an -est (superlative) form "fastest." In addition to any nonsubcategorizable rules triggered by adjectives, "fast" subcategorizes the rules A^4 and A^5. "Fast" can also be a verbal adverb (VAdv), as opposed to a sentential adverb; in this form it does not subcategorize any rules, but does trigger any nonsubcategorized rules calling for verbal adverbs (of which there is in fact only one in the grammar). Note that the adverbial entry also lists -er and -est forms. The duplication is necessary in this case because the adjective and adverb happen to share comparative and superlative forms. Such duplication is not generally true of words having lexical entries for multiple parts of speech.

The lexical entry for "will" in (15) lists two parts of speech: "will" can be an auxiliary verb (Aux), in which case it has the features "^tense present," "aux," and "modal;" or it can be a noun, with the plural form "wills."

4.2. More Complex Lexical Entries: Dynamically Generated Rules

Verbs appear only in subcategorized frames; that is, there is no nonsubcategorized rule which introduces verbs. We have postulated in excess of forty such subcategorized rules. Keeping track of the rules by name (e.g. VP37) became quite difficult, since there was no easy way of making the names mnemonic. We therefore opted to have the lexical entries for verbs contain a listing of the complement structure of the verb, and let the actual rules be generated dynamically as the lexical entries were loaded into the grammar. Consider then the following lexical entry for "see":

```
(16) (word2 see ((V
                  supercategory VP
                  complements (((NP))
                              ((NP) (VP efeatures (ing)))
                              ((NP) (VP efeatures (inf))))
                  inflections ((3s sees)
                              (present see)
                              (past saw)
                              (en seen)
                              (ing seeing)
                              (passive seen
                                    complements ((nil)
                                                ((VP efeatures (ing)))
                                                ((VP  efeatures  (to))))
     )))))
```

The lexical entry is to be read as follows. *Word2* is the name of the macro that processes lexical entries of this form. "See" is a verb (V), and together with its complements forms a "VP" category (the *super-category*). "See" has three possible sets of *complements* in its active forms: a single NP (its direct object), as in "We saw John;" an NP followed by a VP bearing the feature "ing" (i.e. a VP headed by a present participle, as in "We saw John leaving"); and an NP followed by a VP bearing the feature "inf" (i.e. a naked infinitival), as in "I saw John bend the spoon with his fingers."

The lexical entry next lists the *inflections*. All of these (except for the *passive* form) share the properties of the word "see" as already listed, aside from their print form; additionally, each of the tensed forms is assigned the features "^tense present" or "^tense past" as apppropriate, while the past and present participial forms are assigned the features "en" and "ing" respectively. The *passive* form, however, does not share the complement structure of the active forms; instead, the passive's subcategorized complements are listed separately. The first list of complements is "nil," i.e. the passive can be intransitive, as in "John was seen." (An agentive by-phrase is treated as an adverbial modifier.) Additionally, the passive can appear with an *-ing* VP, as in "John was seen leaving," or with a to-VP, as in "John was seen to bend the spoon with his fingers."

Actual lexical entries are somewhat complicated by the fact that there is an extra slot under each complement listing (not shown here) for a rule of semantic interpretation. Lexical entries are also highly commented in the current grammar, a fact which is made easy by the use of the Lisp formalism.

5. Conclusion

We have described a parsing system based on a grammar rule formalism that we believe is easy to use and capable of describing a reasonable subset of natural language. Among the more exotic syntactic constructions handled by the current grammar are:

direct and indirect *wh*-questions
parasitic gap constructions
tough-movement constructions
tensed and infinitival relative clauses (extraposed and nonextraposed)
it-extraposition
comparatives (including comparative subdeletion constructions)
so-called small clause complements
compound nouns
conjunctions of like categories (AP and AP, NP and NP etc.)
across-the-board extractions ("What did Mary see and John hear?")
floating quantifiers
sentential subjects
topicalized constructions
imperatives.

Some constructions which are not handled at present include:

ergatives ("Into the room walked John")
right node raising constructions
gapping constructions
coordination of unlike categories (e.g. NP + AP, as in "John is a hermit and proud of it")

Some of these can be handled by a complex set of phrase structure rules, but we have not tried to construct those rules.

References

[1] J. Bresnan, "Sentence Stress and Syntactic Transformations" *Language*, (1971), Vol 47, pp. 257-281.

[2] J. Bresnan, *The Theory of Complementation in English Syntax*, Garland Publishing Co., (1979).

[3] N. Chomsky, "Barriers" (ms., MIT).

[4] J.M. Gawron, J. King, J. Lamping, E. Loebner, E.A. Paulson, G. Pullum, I.A. Sag, T. Wasow, "Processing English with a Generalized Phrase Structure Grammar", *20th Annual Meeting of the Association for Computational Linguistics*, (1982), pp 74-81.

[5] G. Gazdar, E. Klein, G. Pullum, and I.A. Sag, *Generalized Phrase Structure Grammar*, Basil Blackwell, (1985).

[6] J. Gueron, "On the Syntax and Semantics of PP Extraposition", *Linguistic Inquiry*, (1980), Vol. 11, pp. 637-678.

[7] J. Gueron and R. May, "Extraposition and Logical Form", *Linguistic Inquiry*, (1984), Vol. 15, pp. 1-31.

[8] J. Koster, and R. May, "On the Constituency of Infinitives", *Language*, (1981), Vol. 58, pp. 116-143.

[9] D. Proudian and C. Pollard, "Parsing Head-Driven Phrase Structure Grammar", *23rd Annual Meeting of the Association for Computational Linguistics*, (1985), pp 167-171.

[10] S.G. Pulman, "A Parser That Doesn't", *Second Conference of the European Chapter of the Association for Computational Linguistics*, (1985), pp 128-135.

[11] S. Rosenschein and S. Shieber, "Translating English into Logical Form", *20th Annual Meeting of the Association for Computational Linguistics*, (1982), pp 1-8.

[12] L.K. Schubert and J. Pelletier "From English to Logic: Context-Free Computation of 'Conventional' Logical Translation", *Computational Linguistics*, (1982), Vol. 8 pp. 27-44.

[13] S. Shieber, "Using Restriction to Extend Parsing Algorithms for Complex-Feature-Based Formalisms", *23rd Annual Meeting of the Association for Computational Linguistics*, (1985), pp 145-152.

TRACK:
TOWARD A ROBUST NATURAL LANGUAGE INTERFACE

Sandra Carberry

Department of Computer and Information Science
University of Delaware
Newark, Delaware 19716 USA

ABSTRACT

Analysis of naturally occurring dialogue indicates that human speakers do not process each utterance in isolation but instead use the context in which an utterance occurs to understand it and generate cooperative, helpful responses. The TRACK system is an ongoing research effort aimed at developing a robust interface capable of engaging in natural dialogue with an information-seeker. TRACK assimilates an information-seeking dialogue and uses this accumulated knowledge to understand two classes of utterances that other natural language systems are unable to handle. This paper presents an overview of TRACK, including its components for inferring and modeling an information-seeker's underlying task-related plan and its strategies for handling imperfect and incomplete utterances.

INTRODUCTION

Natural language interfaces to information systems must do more than merely produce direct answers to well-formed queries. Cohen, Perrault, and Allen [6] demonstrated that human question-answerers "expect to engage in a conversation whose coherence is manifested in the interdependence of their often unstated plans and goals with those of the system". Thus to meet the expectations of human users, a natural language system must assimilate the dialogue and use the acquired knowledge to enhance effective communication.

The majority of information-seeking dialogues contain two participants, one seeking information and the other attempting to provide that information. There are many dimensions along which humans glean information from a dialogue, but one of the most significant of these is the inference of the underlying task-related plan motivating an information-seeker's queries. Human information-providers generally attempt to infer this plan and use this acquired knowledge to interpret subsequent utterances and provide useful responses.

We are interested in a class of information-seeking dialogues in which task execution occurs subsequent to the dialogue. This class is representative of a large percentage of interactions with database management systems, decision support systems, and expert systems. Typical tasks include purchasing a home in a real-estate domain and obtaining a degree in a university domain.

(a) This work has been partially supported by a grant from the National Science Foundation, IST-8311400, and a subcontract from Bolt Beranek and Newman Inc. of a grant from the National Science Foundation, IST-8419162

In a cooperative information-seeking dialogue, the information-provider is engaged in helping the information-seeker complete construction of his task-related plan. In order to do so, the information-provider must infer information about that task. It is the job of the information-seeker to communicate whatever information is necessary for the information-provider to fulfill her helping role. However, naturally occurring communication is both imperfect and incomplete. Not only does the information-seeker fail to communicate all aspects of his underlying task and partially constructed plan for accomplishing it, but also his utterances are often imperfectly or incompletely formulated. We find that in naturally occurring dialogues, the information-provider often appears to use her inferred knowledge about this task-related plan to remedy many of the information-seeker's faulty utterances.

One common approach used by natural language interfaces in processing ill-formed queries is to notify the user of errors and possibly present a paraphrase of the system's interpretation for approval. However, an informal poll of a dozen human subjects indicates that users would be unhappy with a system that constantly interrupted the dialogue to point out minor errors or ask if the system's interpretation was correct. Thus a robust natural language interface must be able to handle imperfect and incomplete utterances to which humans respond with relative ease. Otherwise the system will force human information-seekers to concentrate on how to interact with the system instead of on how to solve their problems.

USE OF DERIVED KNOWLEDGE

Human communication is both imperfect and incomplete. An utterance may be ill-formed in the strict syntactic or semantic sense or it may present pragmatic problems in understanding. In addition, humans persist in using abbreviated statements and queries, even in the presence of explicit and repeated instructions to adhere to syntactically and semantically complete sentences [4].

Our model for understanding utterances is based on the Gricean theory of meaning [7]-[8]. According to Grice's theory, a listener must infer a speaker's intent in making an utterance. In an information-seeking dialogue, a cooperative information-provider will assimilate the preceding dialogue, infer the underlying task-related plan motivating the speaker's queries, and focus on that aspect of the task on which the information-seeker's attention is centered. Given an imperfect or incomplete utterance, the listener will be guided by the Gricean maxim of relevance [7] and will use this acquired knowledge to attempt to deduce the speaker's intentions and enable the dialogue to continue without interruption.

Purchase(IS, _res:&CONDOMINIUM)
Preconditions:
  Sale-Status(_res:&CONDOMINIUM, FOR-SALE)
  Satisfy-Restrictions(IS,_res:&CONDOMINIUM)
Actions:
  Negotiate-Price(IS,_res:&CONDOMINIUM)
  Close-Sale(IS, _res:&CONDOMINIUM)
Effects:
  Own(IS, _res:&CONDOMINIUM)

Figure 1. Plan for Purchasing a Condominium

The next sections describe how the TRACK system assimilates an ongoing dialogue and uses the acquired knowledge to understand two classes of problematic utterances. Although the TRACK system employs several knowledge sources, the emphasis in this paper will be on the inferred task-related plan. The information-seeker and information-provider will be referred to as IS and IP respectively.

PLAN INFERENCE

The previous section showed that plan inference was essential to understanding the user's intent. Allen [1],[12] inferred a speaker's goal in seeking gate and time information from an agent in a train setting. However Allen's work was limited to inference in a restricted domain in which the goals could be accomplished by a few primitive steps and a single goal could be assumed to encompass the complete objective of the user. In more complex domains, IS's complete plan consists of a hierarchy of subplans and subgoals which accomplish his overall goal. Such a complete plan is not immediately evident from a single utterance; furthermore, IS's current goal within such a plan changes during the course of a dialogue. Natural language understanding requires that enough of the plan structure be built to represent IS's communicated plans and goals and that the system track the focus of attention in this plan structure.

The TRACK system is able to infer IS's goal in this more complicated situation. For each given domain, TRACK must be provided with the set of possible goals the user may have and a set of plans for achieving these goals. A plan in our system is a hierarchical structure of component goals and actions, each of which has an associated plan or is a primitive in the domain. Figure 1 illustrates a plan that might be used by TRACK. Notice that this plan is hierarchical in that many of the goals and actions contained in the plan may themselves be expanded into plans. For example, the action

Close-Sale(IS, _res:&CONDOMINIUM)

would have an associated plan with constituent actions such as Obtain-Financing and Check-Legal-Documents; this plan could be substituted for the Close-Sale action, thereby expanding in greater detail the plan for purchasing a condominium.

TRACK hypothesizes and tracks IS's changing task-level goals during an information-seeking dialogue. Its processing strategies and heuristics are completely domain-independent and are based on observed characteristics of naturally occurring information-seeking dialogues. Since IS's overall goal and the details of his plan for accomplishing it are not usually evident at the outset of a dialogue, a cooperative information-provider must understand each new utterance in terms of the partial plan she has inferred for IS and the aspect of the task on which she believes IS is currently focused, and use these to appropriately infer additional elements of that plan. TRACK accomplishes this by using focusing heuristics to relate new utterances to the existing inferred plan, called the *context model*, and to expand the context model as the dialogue progresses. Details of TRACK's heuristics are presented in [3].

The previous sections have argued for the need for inferring the information-seeker's task-related plan and have introduced the TRACK system for performing this task. In the following sections, we will show how TRACK uses this information to handle several problematic forms of input.

PRAGMATICALLY ILL-FORMED UTTERANCES

Any system, whether human or machine, must have a model of the world on which it bases understanding. An utterance may be syntactically and semantically well-formed, yet violate the pragmatic rules of the listener's world model. Consider for example the utterance

"Which apartments are for sale?"

In a real-estate world model, single apartments are rented, not sold. However houses, condominiums, apartment buildings, and office buildings are sold. Thus the above utterance contains the erroneous proposition

Sale-Status(_x:&APARTMENT, FOR-SALE)

Analysis of naturally occurring dialogue indicates that utterances violating the listener's world model occur frequently and that human information-providers often modify an information-seeker's ill-formed query to form a similar query that is meaningful and relevant to the established dialogue context. Consider, for example, the following two dialogue sequences:

[1] IS: "I'd like to own my residence but I don't like a lot of maintenance."
"Which apartments are for sale?"

[2] IS: "We'd like to invest between 20 and 30 million dollars."
"Which apartments are for sale?"

The task-related plan inferred as a result of the first utterance in each dialogue sequence affects how a human listener might modify the subsequent query in order to produce one that is both pragmatically correct and relevant to the dialogue. In the first case, the human listener might interpret the utterance as

"Which condominiums are for sale?"

and in the second case as

"Which apartment buildings are for sale?"

We claim that a natural language system must have the ability to respond to such pragmatically ill-formed input as a human conversational partner would.

Other researchers have addressed variations of this problem [5],[11],[13],[15], but each of these analyzed utterances in isolation from the preceding dialogue. Thus they failed to address the speaker's perceived intentions in making an utterance. We claim that the plan inferred for the information-seeker is essential to recognizing and responding to such utterances.

PLAN-BASED RESPONSES

We are interested in being able to respond to utterances that are ill-formed with respect to the system's model of the world. We assume that this model contains entities (objects) and entity sets (collections of entities of the same type)

arranged in a generalization hierarchy, has attributes (properties) associated with the members of an entity set, and has relations that can exist between members of designated entity sets. Utterances that are ill-formed with respect to this model contain a proposition that is erroneous --- that is, one that represents a non-existent attribute or entity set relationship.

TRACK uses IS's inferred task-related plan to suggest modifications to an erroneous proposition contained in an utterance; these changes would make the utterance correct with respect to the system's world model. Only revised queries that might represent IS's intent in making the utterance or at least satisfy his perceived needs are considered.

TRACK uses two classes of substitution heuristics. The first proposes a simple substitution of an attribute, relation, or entity set for that used by the speaker. The second proposes substituting a conjunction of propositions representing an expanded relationship path for the erroneous proposition used by the speaker.

Consider again the dialogue

IS: "I'd like to own my residence but I don't like a lot of maintenance."
"Which apartments are for sale?"

From the first utterance, TRACK would infer that IS wants to purchase a residence with little owner-maintenance; such a plan would include the potential action

Purchase(IS, _res:&CONDOMONIUM)

whose associated plan was shown in Figure 1. IP expects that IS will want to complete instantiation and expansion of his partially constructed plan, and therefore might want to know which entities satisfy the plan propositions. Suppose the context model inferred for IS contains a proposition specifying an attribute Att of a member of entity set Ent-Set2, namely

Att(Ent2:&ENT-SET2, att-value:&ATT-DOMAIN)

and that IS erroneously requests the members of entity set Ent-Set1 with a particular value of Att. Then a cooperative listener might infer that IS wants the members of entity set Ent-Set2 with the indicated attribute value, especially if entity sets Ent-Set1 and Ent-Set2 are semantically similar.

The above analysis is the basis for the following substitution heuristic used by TRACK:

If IS's utterance erroneously presumes that a member Ent1 of entity set Ent-Set1 has an attribute Att, then replace entity set Ent-Set1 with entity set Ent-Set2 if

1. a proposition specifying attribute Att on a member Ent2 of entity set Ent-Set2 appears in an expansion of the context model
2. Ent1 is a variable and the other arguments in IS's proposition unify with the corresponding arguments in the plan proposition (two arguments unify if they are identical constants, variables of the same type, or a constant and a variable of the same type).

IS's second utterance in our example dialogue contains the erroneous proposition

Sale-Status(_x:&APARTMENT, FOR-SALE)

and an expansion of the plan inferred for IS contains the proposition

Sale-Status(_res:&CONDOMINIUM, FOR-SALE).

The above rule would suggest substituting the entity set CONDOMINIUM for the entity set APARTMENT in the erroneous proposition in IS's query, resulting in a revised query of

"Which condominiums are for sale?"

"Which condominiums are for sale?"

If the suggestion mechanism proposes only one revised query, as was the case in this example, then IP is justified in believing that it represents IS's intent since it is the only variant of the erroneous utterance that is relevant to the established dialogue context. If multiple revised queries are proposed, two criteria are important in selecting the most appropriate interpretation. The first is relevance of a revised query to the current focus of attention in the dialogue. The second is the semantic similarity of a revised query to IS's actual utterance. Metrics based on focusing rules and a generalization hierarchy are used to evaluate each suggested revised query. Details of the selection mechanism can be found in [2].

INTERSENTENTIAL ELLIPSIS

Intersentential ellipsis is another kind of input that is difficult for current natural language systems to handle. As illustrated by the example

"I want to cash this check. Small bills only."

intersentential elliptical fragments cannot be fully understood in and of themselves. Therefore a strategy for interpreting such fragments must rely on knowledge obtained from sources other than the fragment itself. Syntactic and semantic strategies have been extensively investigated [4],[10],[16], but the contributions of the speaker's plans and goals to the interpretation of ellipsis has hitherto been inadequately explored.

The power of a plan-based approach is its reliance on pragmatic information, including discourse content and conversational goals, rather than precise representations of the preceding utterances alone. Allen [1] was the first to relate ellipsis processing to the domain-dependent plan underlying a speaker's utterance. However Allen's restricted train domain contained only two overall goals with associated plans, and these task-related plans were simple, single-level structures completely built-in and considered only as single entities. In more complex domains, it is necessary to identify the speaker's discourse (communicative) goal and the particular aspect of the speaker's overall task-related plan addressed by the elliptical fragment in order to interpret it properly.

A speaker can felicitously employ intersentential ellipsis only if he believes his utterance will be properly understood. TRACK's motivating hypothesis is that IS and IP mutually believe that certain knowledge has been acquired during the course of the dialogue and that this factual knowledge along with other processing knowledge will be used to deduce IS's intentions. The requisite factual knowledge includes IS's inferred task-related plan (normally only a partial plan during the dialogue), IP's beliefs about IS's beliefs, and the anticipated discourse goals of IS. The requisite processing knowledge includes plan recognition strategies and focusing techniques. In this paper, we shall address only two of these knowledge sources: the inferred task-related plan and focusing techniques.

IS's inferred task-related plan provides the context within which an elliptical utterance should be understood. Focusing techniques are necessary in order to identify that portion of the underlying plan to which a fragmentary utterance refers. Consider for example the following dialogue sequence:

IS: "I want to register for a course."
"But I missed pre-registration."
"The cost?"

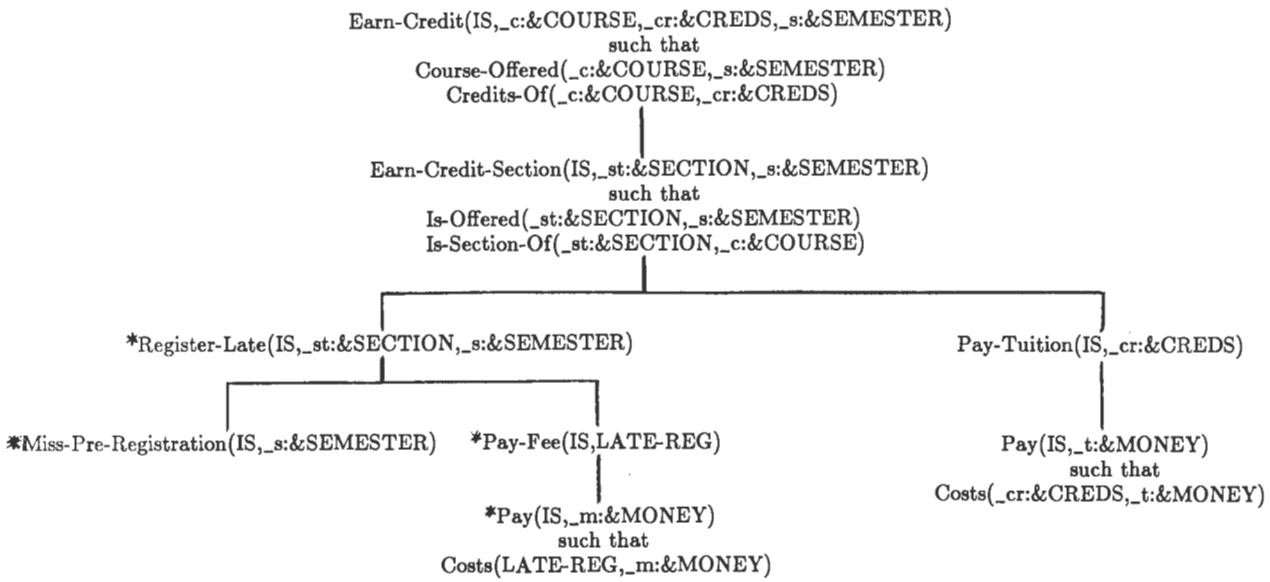


Figure 2. Portion of an Expanded Context Model

IS's underlying task is to take a course; a plan for accomplishing this task will include several "cost" attributes, namely the cost of the course, the cost of the textbooks, the cost of any lab fees, and in this case, the cost of late-registering. If one wants to register for a course and misses pre-registration, then one must late-register; the utterance preceding the elliptical fragment establishes late-registering as the current focus of attention in the dialogue and as a result, the elliptical fragment is interpreted as a request for the cost of the extra fees involved in late-registration.

ELLIPSIS INTERPRETATION FRAMEWORK

If a sentence fragment is detected, TRACK initiates ellipsis processing. Any model of a preceding dialogue will contain a representation of IS's underlying task-related plan, a discourse stack of anticipated discourse goals such as "Answer-Question", and a model of inferred beliefs. Rules for constructing the anticipated discourse goals for simple dialogues can be found in [2].

Our ellipsis processing framework is a top-down strategy that uses IS's anticipated discourse goals to guide interpretation of the fragment and relate it to the underlying task-related plan. The discourse component first applies discourse expectation rules to the top element of the discourse stack and suggests one or more potential discourse goals which IS might be expected to pursue.

The analysis component uses the context model and a belief model to suggest possible associations of the elliptical fragment with aspects of IS's task-related plan. If multiple associations are suggested, the evaluation component applies focusing strategies to the context model to select the association believed intended by IS --- namely, that most appropriate to the current focus of attention in the dialogue.

The discourse component uses discourse goal rules and the results produced by the analysis component to determine if the fragment accomplishes the proposed discourse goal; if so, the ellipsis processor interprets the fragment relative to that discourse goal. If the discourse component determines that the fragment does not accomplish any of the proposed discourse goals, then the discourse stack is popped and the interpretation process repeated for the fragment and the new top element of the discourse stack.

ELLIPSIS PROCESSING --- AN EXAMPLE

The following is a simple example illustrating our overall strategy for processing ellipsis.

IS: "I want to register for a course."
"But I missed pre-registration."
"The cost?"

The discourse stack immediately prior to the elliptical fragment is

PROVIDE-FOR-ASSIMILATION

indicating that the strongest expectation is for IS to continue providing background information to the system.

The applicable discourse expectation rule suggests the discourse goals of 1)providing further information for assimilation and 2)seeking information in order to construct the task-related plan. The utterance terminates in a "?", ruling out provide for assimilation.

Figure 2 presents a portion of IS's underlying task-related plan inferred from the utterances preceding the elliptical fragment. The semantic representation of IS's elliptical fragment is

_cst1:&MONEY
such that
Costs(_x:&UNKNOWN,_cst1:&MONEY)

The term and proposition in the fragment associate with several terms and propositions in IS's underlying plan, including

_t:&MONEY
[1] such that
Costs(_cr:&CREDS,_t:&MONEY)

_m:&MONEY
[2] such that
Costs(LATE-REG,_m:&MONEY)

appearing in Figure 2 and terms and propositions appearing in portions of the plan not displayed in Figure 2, such as the cost of textbooks.

The evaluation component must select from among these possible associations. We use the notion of focus domains in order to group finely grained actions and associated plans into more general related structures. A focus domain consists of a set of actions, one of which is an ancestor of all other actions in the focus domain and is called the root of the focus domain. If an action is a member of a focus domain and that action is not the root of another focus domain, then all the actions contained in the plan associated with the first action are also members of the focus domain. The use of focus domains allows the grouping together of those actions that appear to be at approximately the same level of implicit focus when a plan is explicitly focused. Nodes preceded by an asterisk in Figure 2 are in the same focus domain.

The current focused plan immediately prior to the elliptical utterance is

Register-Late(IS,_st:&SECTION,_s:&SEMESTER)

and the most recently considered action or goal is

Miss-Pre-Registration(IS,_s:&SEMESTER)

The ellipsis processor employs a set of seven focusing heuristics to select the interpretation intended by IS. The current focus domain contains those actions that are most highly focused within IS's underlying task-related plan; therefore interpretations relevant to these actions are preferred. In our example, the only association within the current focus domain is with

_m:&MONEY
such that
Costs(LATE-REG,_m:&MONEY)

The discourse goal rule associated with seeking-information uses the results of the analysis and evaluation components and finds that the fragment can be interpreted as a request for information in order to further construct the task-related plan. In particular, TRACK deduces that IS is requesting the cost of the fee for late registration.

SUMMARY

Participation in a cooperative dialogue requires far more than merely providing direct answers to perfectly well-formed and complete queries. The information-seeker expects the information-provider to assimilate the dialogue and use this knowledge to understand subsequent utterances and provide helpful responses. This is the objective of the TRACK system, which has been implemented for a university domain containing information on courses, policies, and degree requirements.

We claim that the information-seeker's underlying task-related plan as inferred from the preceding dialogue must be a major component of any interpretation strategy, since this plan captures the perspective from which the information-seeker has made an utterance. Our pragmatics-based strategy is superior to previous approaches because it uses a model of the established dialogue context to identify and address the speaker's perceived intentions in making an utterance.

ACKNOWLEDGEMENTS

I would like to thank Kathy McCoy for her many detailed comments and suggestions on this paper.

REFERENCES

[1] Allen, J.F., "Analyzing Intention in Utterances", *Artificial Intelligence 15(3)*, 1980

[2] Carberry, S., *Pragmatic Modeling in Information System Interfaces*, Ph.D. Dissertation, Dept. of Computer Science, Univ. of Delaware, 1985

[3] Carberry, S., "Tracking User Goals in an Information-Seeking Environment", *Proc. of the Nat. Conf. on Artificial* Intelligence, 1983

[4] Carbonell, J. G., "Discourse Pragmatics and Ellipsis Resolution in Task-Oriented Natural Language Interfaces", *Proc. of the 21rst Annual Meeting of the ACL*, 1983

[5] Chang, C.L., "Finding Missing Joins for Incomplete Queries in Relational Data Bases", Technical Report RJ2145, IBM Research Laboratory, 1978

[6] Cohen, P.R., Perrault, R., and Allen, J.A., "Beyond Question Answering", *Strategies for Natural Language Processing*, Lehnert, W. and Ringle, M., editors, Lawrence Erlbaum Associates, 1981

[7] Grice, H.P., "Logic and Conversation", In *Syntax and Semantics*, Cole and Morgan, editors, Academic Press, 1975

[8] Grice, H.P., "Utterer's Meaning and Intentions", *Philosophical Review 68*, 1969

[9] Grosz, B., "Focusing and Description in Natural Language Dialogues", in *Elements of Discourse Understanding*, Joshi, A., Webber, B., and Sag, I., editors, Cambridge University Press, 1981

[10] Hendrix, C. G., Sacerdoti, E.D., and Slocum, J., "Developing a Natural Language Interface to Complex Data", Technical Report, SRI International, 1976

[11] Kaplan, S.J., "Cooperative Responses from a Portable Natural Language Query System", *Artificial Intelligence 19*, 1982

[12] Litman, D.J. and Allen, J.F., "A Plan Recognition Model for Clarification Subdialogues", *Proc. of the Int. Conf. on Computational Linguistics*, 1984

[13] Mays, E., "Failures in Natural Language Systems: Applications to Data Base Query Systems", *Proc. of the Nat. Conf. on Artificial Intelligence*, 1980

[14] Sidner, C.L., "Plan Parsing for Intended Recognition in Discourse", *Computational Intelligence*, vol. 1, 1985

[15] Sowa, J.F., "Conceptual Graphs for a Data Base Interface", *IBM Journal of Res. and Dev.*, 1976

[16] Weischedel, R.M. and Sondheimer, N.K., "An Improved Heuristic for Ellipsis Processing", *Proc. of the 20th Annual Meeting of the ACL*, 1982

Representation of Negative and Incomplete Information in Prolog

Kwok Hung Chan

Department of Philosophy
The University of Western Ontario
London, Ontario, Canada N6A 3K7

Abstract

Logic programs are extended to mixed programs. In a mixed program negative literals are allowed in the heads of clauses. Thus the falsity conditions of predicates can be defined explicitly in a mixed program. This is very useful for open problem domains involving incomplete information or concepts which are conditionally defined.

Mixed programs are interpreted by an extension of SLDNF-resolution, called *Mixed SLDNF-resolution*. A Prolog program implementing Mixed SLDNF-resolution is presented. An approach for decreasing the degree of incompleteness of Mixed SLDNF-resolution is suggested.

1. Negation as Failure

It is well known that negative information is not entailed by definite clause logic programs [8]. For this reason, practical logic programming systems, like Prolog, adopt the rule of negation as failure: Clauses in a program are regarded as *complete definitions* for the predicates in the heads of clauses. Although biconditionals are not explicitly available in the language, conditional clauses are taken to represent biconditionals implicitly. A completed program, abbreviated as *comp*(S), is associated with each program S in clausal form. Every predicate symbol in S has a (biconditional) definition in comp(S). A predicate symbol occurring only in the bodies of the clauses is assigned the empty set as its extension. And clauses with the same predicate symbol in their heads are blended into one biconditional definition of the predicate in question [5]. SLD-resolution[1] is extended to SLDNF-resolution: SLD-resolution augmented by the rule of negation as failure ([5], [1], [8]). For definite clause logic programs, SLDNF-resolution incorporating a safe selection rule[2] is both sound and complete: A goal G is a logical consequence of comp(S) if and only if $S \cup \{G\}$ has a finitely failed SLD-tree.[3]

Using the rule of negation as failure, it is possible to extend logic programs to general programs, where a general program is one which may have negative literals in the body of its program clauses. Negative literals are also allowed in goal clauses [8].

The rule of negation as failure is very useful for logic programs that describe problem domains in which the closed world assumption is true ([6], [10]). For such a closed problem domain D, we can have a logic program S that captures the truth and falsity of every predicate. Accordingly a ground atom A is true if and only if A is implied by S. Syntactic relations are typical examples of such closed problem domains. For example, the list-membership relation can be completely defined by the following familiar program.

S1: $member(h,[h|t]) \leftarrow$
$member(x,[h|t]) \leftarrow member(x,t)$

A term t is a member of a list L if and only if $member(t,L)$ is a logical consequence of *S1*.

2. Open Problem Domains

Real world situations are typically *open problem domains*: A problem domain D is said to be open if we do not have complete information about the extensions of relations in D. For example, given an object and a

[1] SLD-resolution is Linear resolution with Selection rule for Definite clauses. See [7], [1] or [8].

[2] A selection rule is safe if and only if a negative literal may be selected only if it is ground [8].

[3] This paper assumes that the reader is familiar with the negation as failure rule [8].

property, we may not know whether the object has the property.

The rule of negation as failure is too strong for logic programs that describe open problem domains. For example, John does not know where Mary was last Sunday. Should a logic program *S2* which encodes John's knowledge imply information about where Mary was last Sunday? Suppose John is a detective investigating a murder that took place last Sunday in a club and Mary is one of the suspects. It is very important that the logic programming system that John uses should not infer the falsity of p(="Mary was in the club last Sunday") simply because it fails to prove p from *S2*. However, if John uses a system implementing the negation as failure rule, the system will infer that p is false precisely because it fails to prove p from *S2*.

On the other hand, John may know that Mary was not home last Sunday, and this is an important piece of information. John should be able to encode this negative information in a logic program without encoding positive information about Mary's whereabouts. Thus logic programs encoding knowledge about an open problem domain should be able to represent negative information explicitly. Rather than specify the conditions under which a relation holds, and assume by default that the relation does not hold whenever the conditions are not satisfied, we should be able to state both the truth conditions and the falsity conditions of a relation explicitly. This can be accomplished by further extending logic programs to *open programs*, where an open program is one which may have negative literals in the bodies as well as in the heads of its program clauses. If *S2* is an open program, it may contain the following clauses:

$$location(Tom, 5{-}18{-}86, club) \leftarrow$$
$$\neg location(Mary, 5{-}18{-}86, Mary's{-}home) \leftarrow$$

3. Conditionally Defined Concepts

Open programs are also needed for domains involving terms which are conditionally defined. A predicate P is said to be conditionally defined if and only if P or $\neg P$ apply only to n-tuples satisfying certain preconditions. If an n-tuple $\vec{s}$ does not satisfy the preconditions, neither $P(\vec{s})$ nor $\neg P(\vec{s})$ is true. Conditionally defined terms are commonly encountered. Consider, for example, a test on the effectiveness of a new drug K. Patients are divided into two groups: the experimental group and the control group. Patients in the experimental group take the drug K, while patients in the control group take a placebo. If a patient x takes drug K and shows improvement, K is effective for x. If x shows no improvement, K is ineffective for x. However, for a patient y in the control group, because y has not taken the drug K, K is neither effective nor ineffective for y. Let $effective(x)$ mean drug K is effective for x, $drug(x)$ mean x has taken drug K, and $improvement(x)$ means x has shown improvement. Then *effective* may be defined by the following pair of open clauses:

$$effective(x) \leftarrow drug(x), improvement(x)$$
$$\neg effective(x) \leftarrow drug(x), \neg improvement(x)$$

And *effective* is undefined for patients who have not taken drug K.[4]

4. Mixed programs

Because an open logic program uses predicates like *member* which are completely defined, it is necessary to develop logic programming systems to interpret logic programs containing both completely and partially defined predicates, called *mixed logic programs*.

The predicates can be partitioned into completely defined and partially defined predicates. Based on this partition a mixed program is partitioned into three subprograms: (i) a general program, S_c, containing definitions of completely defined predicates; (ii) an open program, S_o, containing definitions of partially defined predicates; (iii) a program, S_2, containing information about the partition of the predicates.

S_c contains clauses that define completely defined predicates. These completely defined predicates must be defined in terms of completely defined predicates only. Predicates that are partially defined may not be used to define predicates which are completely defined. Accordingly, S_c forms an independent unit to which SLDNF-resolution can be applied.

S_o contains clauses that define partially defined predicates. Open program clauses of the form

$$P(\vec{s}) \leftarrow L_1, \ldots, L_n$$

define the truth conditions of the predicate P; while open program clauses of the form

$$\neg P(\vec{s}) \leftarrow L_1, \ldots, L_n$$

define the falsity conditions of P. Any predicate, whether partially or completely defined, may be used to define a partially defined predicate.

[4] The predicate *effective* is better analysed as a disposition term, a modal notion. However, if we are considering properties such as this in a first order language we need to use conditional definitions ([2], [3]).

An open program may be updated[5] by adding clauses, by so doing the extensions of partially defined predicates and their complements[6] may be extended. If we had allowed completely defined predicates to be defined in terms of partially defined predicates, then the extension of completely defined predicates will change as we update the extensions of partially defined predicates.

S_2 contains clauses of the form *completely–defined*(P), where *completely-defined* is a second level predicate introduced to encode the information about the partition of predicates. For every predicate P in the program, there is a clause *completely–defined*(P) in S_2 if and only if P is completely defined.[7] Information in S_2 is used by the interpreter to determine if a predicate is completely defined.

The program TEST below is an example of a mixed program. The predicates *drug*, *improvement*, and *placebo* are completely defined; while the predicate *effective* is partially defined. Definitions for the completely defined predicates form a subset S_c, while those for partially defined predicates form another subset S_o. The subset S_2 contains clauses of the form *completely–defined*(P), one for each completely defined predicate.

TEST:

S_2: *completely–defined*(*drug*) ←
completely–defined(*placebo*) ←
completely–defined(*improvement*) ←

S_c: *drug*(*a*) ←
drug(*b*) ←

improvement(*a*) ←
improvement(*c*) ←

placebo(X) ← *not*(*drug*(X))

S_o: *effective*(X) ←
drug(X), *improvement*(X)

not(*effective*(X)) ←
drug(X), *not*(*improvement*(X))

The negation of a literal L is formed by prefixing 'not' to (L). For example, the negation of *drug*(a) is *not*(*drug*(a)). Note that 'not' is not a predicate symbol, but rather a Prolog functor used as a sentential connective.

5. Mixed SLDNF-Resolution

In this section, we introduce an extension of SLDNF-resolution, called *Mixed SLDNF-resolution*, for interpreting mixed programs. Mixed SLDNF-resolution applies SLD-resolution to partially defined predicates, and SLDNF-resolution to completely defined predicates.

The notion of a *safe selection rule* is extended: if a completely defined predicate P occurs in a negative literal L of a goal clause, a safe selection rule may select L only if L is ground.

Mixed SLDNF-resolution works as follows: Positive subgoals are proved using SLD-resolution. When a ground negative literal $\neg A$ with a completely defined predicate is selected, an attempt is made to prove A and the result is interpreted according to the negation as failure rule. If a negative literal $\neg P(\vec{s})$, where P is a partially defined predicate, is selected, $\neg P(\vec{s})$ is proved using clauses in S_o with $\neg P$ in their heads.

The notion of a *correct answer substitution* is extended as follows: Let $S=S_2 \cup S_c \cup S_o$ be a mixed program, $G(=\leftarrow(A_1, \ldots, A_n))$ a general goal, and θ a substitution for variables of G. θ is called a *correct answer substitution* for $S_o \cup comp(S_c) \cup \{G\}$ if and only if $\forall((A_1 \wedge \cdots \wedge A_n)\theta)$[8] is a logical consequence of $S_o \cup comp(S_c)$.

Mixed SLDNF-resolution is formally defined and its soundness proved in the full version of the paper [4].

6. Implementation in Prolog

In this section, a Prolog program MSLDNF that implements Mixed SLDNF-resolution is presented. A Prolog system with MSLDNF in its database is called *Mixed Prolog*.

---

[5] Here *updating* refers to the operation of adding new clauses without deleting old ones.

[6] The extension of a predicate P is defined here as the set of tuples $\vec{s}$ such that $P(\vec{s})$ is (known to be) true; while the extension of the complement of P is defined as the set of tuples $\vec{s}$ such that $P(\vec{s})$ is (known to be) false.

[7] It is assumed that predicate symbols with different arities have different names. The system can be extended easily to permit sharing of name by predicate symbols of different arities.

[8] $\forall(C)$ is the universal closure of C.

MSLDNF:

(a): $not(A) \leftarrow$
$A = . .[P \mid Argument\text{-}list]$,
$completely\text{-}defined(P)$,
!,
$ground\text{-}list(Argument\text{-}list)$,
$not\text{-}provable(A)$

$not\text{-}provable(A) \leftarrow$
$call(A)$, !, $fail$.
$not\text{-}provable(A) \leftarrow$

The intended interpretation of $ground\text{-}list(L)$ is: the argument list L is ground. The definition of *ground-list* is omitted in this paper.

To use the program MSLDNF, we consult the file containing MSLDNF *before* we consult other file(s) containing mixed program(s). Then we can type in our queries in the usual way.

We have to consult MSLDNF before we consult other open programs, in order to ensure that clauses in an open program defining the falsity conditions of partially defined predicates come after clause (a). Accordingly, clause (a) is the first input clause selected to be resolved with a subgoal of the form $\leftarrow not(P(\vec{s}))$. When Mixed Prolog attempts to refute $\leftarrow not(P(\vec{s}))$, clause (a) is always selected. It then checks to see if P is completely defined. If it is, Mixed Prolog will not attempt to resolve $\leftarrow not(P(\vec{s}))$ with other clauses in the database. Rather, Mixed Prolog will attempt to refute $\leftarrow not(P(\vec{s}))$ by the negation as failure rule. If P is partially defined, then Mixed Prolog attempts to refute $\leftarrow not(P(\vec{s}))$ using the clauses in the open program(s) that define the falsity conditions of P. Subgoals of the form $\leftarrow P(\vec{s})$ are treated in the standard way.

Positive responses from Mixed Prolog are interpreted in the usual way. Because there are partially defined predicates in a mixed program, negative responses from Mixed Prolog cannot be interpreted by the negation as failure rule: The answer *no* to a query $Q(=A_1, \ldots, A_n)$ only means that Mixed Prolog has failed to refute $\leftarrow(A_1, \ldots, A_n)$. If we desire to verify that a literal L is false, we need to type in the query $not(L)$. If the answer to $not(L)$ is *yes*, then $not(L)$ is a logical consequence of the program. For example, Mixed Prolog's answers to the queries $effective(c)$ and $not(effective(c))$ are *no*. These are correct answers. Because c has not taken drug K, K is neither effective nor ineffective for c.

7. Incompleteness and Inconsistency

Mixed SLDNF-resolution is incomplete. For example, the answer computed by Mixed Prolog for $\text{TEST} \cup \{\leftarrow placebo(X)\}$ and for $\text{TEST} \cup \{\leftarrow not(placebo(X))\}$ are both *no*, although $\exists(placebo(X))$ and $\exists(not(placebo(X)))$ are logical consequences of TEST. This sort of incompleteness is caused by the requirement that the selection rule must be safe. Any attempt to refute a goal of the form $\leftarrow not(P(\vec{s}))$, where P is completely defined and $\vec{s}$ is not ground, fails. This problem of Mixed SLDNF-resolution is inherited from SLDNF-resolution.[9] Since Mixed SLDNF-resolution uses SLDNF-resolution, it inherits the incompleteness of SLDNF-resolution.

There are other sorts of incompleteness introduced by open programs: Allowing both positive and negative literals in the heads of open program clauses causes incompleteness. For example, $\neg Q(a)$ is a logical consequence of the following open program *S3*.

S3: $P(a) \leftarrow$
$\neg P(a) \leftarrow Q(a)$

However, attempts to refute $\leftarrow(\neg Q(a))$ will fail since $\neg Q(a)$ cannot be unified with any literal in the heads of the program clauses. This problem is partially solved by adding logically redundant clauses to the program. The clause

C: $\neg Q(a) \leftarrow P(a)$

is logically equivalent to the last clause in *S3*. However, $\neg Q(a)$ is now derivable from the program $S3 \cup \{C\}$. Adding logically redundant clauses to an open program in this way is tantamount to a selective relaxation of the rule that only one literal in a program clause may be resolved upon. We could further extend Mixed SLDNF-resolution by adding facilities that add such logically redundant clauses when needed.

However, we must not supplement an open program by a logically redundant clause with a completely defined predicate in its head. If the body of the added clause contains a partially defined predicate, the resulting program is not a mixed program. Even if no partially defined predicates occur in the added clause, the intended interpretation of the completely defined predicates may be changed by the addition of the new clause. Consider the following logically equivalent general programs *G1* and *G2*.

G1: $P(a) \leftarrow \neg Q(a)$
G2: $P(a) \leftarrow \neg Q(a)$
$Q(a) \leftarrow \neg P(a)$

[9] See Lloyd [8] for a partial solution to this problem.

$\neg Q(a)$ is a logical consequence of $comp(G1)$, but not $comp(G2)$. Therefore $comp(G1)$ is *not* logically equivalent to $comp(G2)$.

As an open program clause is just a *clause* with a distinguished literal — the head of the clause, open programs have the same *expressive power* as full first order logic.[10] It is, however, difficult to extend Mixed Prolog to a complete refutation system without major modification of Prolog.

Unlike definite clause programs, a mixed program may be inconsistent. Program $S4$ below is an inconsistent open program.

$S4$: $P(a) \leftarrow$
$\neg P(a) \leftarrow$
$\neg Q(a) \leftarrow$

However, because Mixed SLDNF-resolution is incomplete, not every literal is derivable from an inconsistent program. $Q(a)$, for example, is not provable from $S4$.

8. Conclusion

In this paper the concepts of mixed programs and Mixed SLDNF-resolution have been introduced. The usefulness of mixed programs for open problem domains has been illustrated.

Acknowledgement

The author would like to thank E. Elcock, W. Demopoulos, E. Stabler (Jr.), R. Mercer and P. Hoddinott for rewarding discussions.

[10] See [9] section 1.5 for the standard procedure for converting a formula in a first order language to its clausal form.

References

[1] Apt, K. R. and van Emden, M. H.
"Contributions to the Theory of Logic programming".
Journal of the Association for Computing Machinery 29(3):841-862, July, 1982.

[2] Carnap, R.
"Testability and Meaning".
Philosophy of Science 3 & 4, 1936 & 1937.

[3] Carnap, R.
"The Methodological Character of Theoretical Concepts".
Minnesota Studies in the Philosophy of Science.
University of Minnesota Press, 1956, pages 38-76.

[4] Chan, K. H.
"Representation of Negative and Incomplete Information in Prolog".
1986.

[5] Clark, Keith L.
"Negation as Failure".
Logic and DataBases.
Plenum, 1978, pages 293-322.

[6] Gallaire, H. and J. Minker (editors).
Logic and Databases.
Plenum Press, New York, 1978.

[7] Kowalski, R. A. and Kuehner, D.
"Linear Resolution with Selection Function".
Artificial Intelligence 2:227-260, 1971.

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

[9] Loveland, D. W.
Automated Theorem Proving: A logical Basis.
North-Holland, New York, 1978.

[10] Shepherdson, John C.
"Negation as Failure: A comparsion of Clark's Completed Data Base and Reiter's Closed World Assumption".
The Journal of Logic Programming 1(1):51-79, June, 1984.

ON THE LOGIC OF REPRESENTING DEPENDENCIES BY GRAPHS*

Judea Pearl
Computer Science Department
University of California, Los Angeles
Los Angeles, California 90024

and

Azaria Paz
Computer Science Department
Technion, Israel Institute of Technology
Haifa, Israel

ABSTRACT: We consider 3-place relations $I(x, z, y)$ where, x, y, and z are sets of propositional variables, and $I(x, z, y)$ stands for the statement: "Knowing z renders x independent of y." We give sufficient conditions on I for the existence of a (minimal) graph G such that $I(x, z, y)$ can be validated by testing whether z separates x from y in G. These conditions define a GRAPHOID. The theory of graphoids uncovers the axiomatic basis of informational dependencies and ties it to vertex-separation conditions in graphs. The defining axioms can also be viewed as inference rules for deducing which propositions are relevant to each other, given a certain state of knowledge.

1. INTRODUCTION

Any system that reasons about knowledge and beliefs must make use of information about dependencies and relevancies. If we have acquired a body of knowledge z and now wish to assess the truth of proposition x, it is important to know whether it would be worthwhile to consult another proposition y, which is not in z. In other words, before we examine y, we need to know if its truth value can potentially generate new information relative to x, information not available from z. For example, in trying to predict whether I am going to be late for a meeting, it is normally a good idea to ask somebody on the street for the time. However, once I establish the precise time by listening to the radio, asking people for the time becomes superfluous and their responses would be irrelevant. Similarly, knowing the color of x's car normally tells me nothing about the color of Y's. However, if X were to tell me that he almost mistook Y's car for his own, the two pieces of information become relevant to each other -- whatever I learn about the color of X's car will have bearing on what I believe the color of Y's car to be. What logic would facilitate this type of reasoning?

In probability theory, the notion of relevance is given precise quantitative underpinning using the device of *conditional independence*. A variable x is said to be independent of y given the information z if $P(x, y \mid z) = P(x \mid z)\, P(y \mid z)$. However, it is rather unreasonable to expect people or machines to resort to numerical verification of equalities in order to extract relevance information. The ease and conviction with which people detect relevance relationships strongly suggest that such information is readily available from the organizational structure of human memory, not from numerical values assigned to its components. Accordingly, it would be interesting to explore how assertions about relevance can be inferred qualitatively from various models of memory and, in particular, whether the logic of such assertions can be characterized axiomatically.

Since models of human knowledge are often portrayed in terms of various associational networks (e.g. semantic networks [Woods 1975], constraint networks [Montanari 1974] inference nets [Duda, Hart and Nilsson 1976]), a natural starting point would be to examine what types of dependency relations can be captured by a network representation, in the sense that all assertions about dependencies (and independencies) in a given model be deducible from the topological properties of some network.

When we deal with a phenomenon where the notion of neighborhood or connectedness is explicit (e.g., family relations, electronic circuits, communication networks, etc.), we have no problem configuring a graph which represents the main features of the phenomenon. However, in modeling conceptual relations such as causation, association and relevance, it is often hard to distinguish direct neighbors from indirect neighbors; so, the task of constructing a graph representation then becomes more delicate. The notion of conditional independence in probability theory is a perfect example of such a relational structure. For a given probability distribution P and any three variables x, y, z, while it is fairly easy to verify whether knowing z renders x independent of y, P does not dictate which variables should be regarded as direct neighbors. Thus, many topologies might be used to display the dependencies embodied in P.

This paper studies the feasibility of devising graphical representations for dependency models in which the notion of neighborhood is not specified in advance. Rather, what is given explicitly is the relation of "in betweenness." In other words, we are given the means to test whether any given subset S of elements *intervenes in a relation between* elements x and y, but it remains up to us to decide how to connect the elements together in a graph that accounts for these interventions.

Section 1 uncovers the axiomatic basis of dependency models which are isomorphic to vertex separation in graphs. The axioms established can be used both for testing whether a given model lends itself to a complete graphical representation, and for inferring new dependencies from a given initial set. Section 2 examines dependency models called graphoids which may have no graph isomorphism yet possess an effective graphical representation; all their dependencies together with the highest possible number of indepen-

* This work was supported in part by the National Science Foundation, Grants DCR 83-13875 & 85-01234.

dencies can be displayed by some graph. The theory of graphoids, of which probabilistic dependence is a special case, provides methods for constructing such optimal graphs.

2. GRAPH REPRESENTATION - SEMANTICS AND SYNTAX

2.1 What's in a Link? Suppose we have a collection U of interacting elements and we decide to represent their interactions by an undirected graph G in which the nodes correspond to individual elements of U. Naturally, we would like to display independence between elements by the lack of connectivity between their corresponding nodes in G and, conversely, dependent elements should correspond to connected nodes in G. This requirement alone, however, does not take full advantage of the expressive power of graph representation. It treats all connected components of G as equivalence classes and does not attribute any special significance to the topological configuration within each connected component of G.

Clearly, if graph topology is to convey meaning beyond its connectedness, a semantic distinction must be made between "direct connection" and "indirect connection" in the sense that arbitrarily adding a link between otherwise connected elements should correspond to a totally different state of dependency. This means that the model which governs our understanding of the interactions between the elements of U must also provide us with a criterion for testing 3-place, *conditional independence* relations of the form $I(x,z,y)$ = "x is independent of y conditioned on z." While a variety of interpretations might be given to the terms "independent" and "conditioned on," we shall see that some reasonable general constraints can be imposed on the relation $I(x,z,y)$ if we associate it with the intuitive statement: "knowing y would tell me nothing new about x, if I already know z."

Ideally, we would like to require that if the removal of some subset S of nodes from the graph G renders nodes x and y disconnected (written $< x \mid S \mid y >_G$), then this separation should correspond to conditional independence between x and y given S, namely,

$$< x \mid S \mid y >_G \Longrightarrow I(x,S,y)$$

and, conversely,

$$I(x,S,y) \Longrightarrow < x \mid S \mid y >_G$$

This would provide a clear graphical representation for the notion that x does not affect y directly, that its influence is mediated by the variables in S. Unfortunately, we shall next see that these two requirements are too strong; there is often no way of using vertex separation in a graph to display *all* dependencies and independencies embodied in the common notion of information relevancy.

Let $U = \{\alpha, \beta, \ldots\}$ be a finite set of elements (e.g. propositions, variables etc.) and let x, y, and z stand for three non intersecting subsets of elements in U. Let M be a model which assigns truth values to the 3-place predicate $I(x,z,y)$ or, in other words, M determines a subset I of triplets (x,y,z) for which the assertion "x is independent of y given z" is true.

Definition: An undirected graph G is a *dependency map* (D-map) of M if there is a one-to-one correspondence between the variables in U and the nodes of G, such that for all non-intersecting subsets, x, y, z, of variables we have:

$$I(x,z,y)_M \Longrightarrow < x \mid z \mid y >_G \qquad (1)$$

Similarly, G is an *Independency map* (I-map) of M if:

$$I(x,z,y)_M \Longleftarrow < x \mid z \mid y >_G \qquad (2)$$

A D-map guarantees that vertices found to be connected are, indeed, dependent, but it may occasionally display dependent variables as separated vertices. An I-map works the opposite way: it guarantees that vertices found to be separated always correspond to genuinely independent variables but does not guarantee that all those shown to be connected are, in fact, dependent. Empty graphs are trivial D-maps, while complete graphs are trivial I-maps.

It is not hard to see that in many reasonable models of informational dependency no graph can be both a D-map and and I-map of M. For example, in models where $I(x,z,y)$ means "y is irrelevant to x once we learn z", we often find *nonmonotonic* behavior -- totally unrelated propositions can become relevant to each other upon learning new facts. For instance, whether it is cloudy or sunshine outside has nothing to do with the type of paper I am currently writing on. However, upon learning that I have difficulty reading my pencil marks, seeing the sun shining through the window makes me doubt the quality of the paper. Such a nonmonotonic model M, implying both $I(x,z_1,y)_M$ and $NOT\text{-}I(x,z_1 \cup z_2, y)_M$, cannot have a graph representation which is both an I-map and a D-map, because graph separation always satisfies $< x \mid z_1 \mid y >_G \Longrightarrow < x \mid z_1 \cup z_2 \mid y >_G$ for any two subsets z_1 and z_2 of vertices. Thus, D-mapness forces G to display z_1 as a cutset separating x and y, while I-mapness prevents $z_1 \cup z_2$ from separating x and y. No graph can satisfy these two requirements simultaneously.

Being unable to provide graphical representations to some (e.g. nonmonotonic) interpretations of $I(x,z,y)$, raises the question of whether we can formally delineate the class of models which *do* lend themselves to graphical representation. This is accomplished in the following substitution by establishing an axiomatic characterization of the type of dependency relations which are isomorphic to vertex separation in graphs.

2.2 Axiomatic Characterization of Graph-Isomorph Dependencies

Definition: A dependency model M is said to be *graph-isomorph* if there exists a graph $G = (U, E)$ which is both an I-map and a D-map of M, i.e., for every three non-intersecting subsets x, y and z of U we have:

$$I(x,z,y)_M \Longleftrightarrow < x \mid z \mid y >_G \qquad (3)$$

Theorem 1: A necessary and sufficient condition for a dependency model M to be graph-isomorph is that $I(x,z,y)_M$ satisfies the following five independent axioms (the subscript M dropped for clarity):

(symmetry)

$$I(x,z,y) \Longleftrightarrow I(y,z,x) \qquad (4.a)$$

(subset closure)

$$I(x,z,y \cup w) \Longrightarrow I(x,z,y) \;\&\; I(x,z,w) \qquad (4.b)$$

(intersection)

$$I(x, z \cup w, y) \;\&\; I(x, z \cup y, w) \Longrightarrow I(x, z, y \cup w) \qquad (4.c)$$

(strong union)

$$I(x, z, y) \Longrightarrow I(x, z \cup w, y) \quad \forall w \subset U \qquad (4.d)$$

(transitivity)

$$I(x, z, y) \Longrightarrow I(x, z, \gamma) \text{ or } I(\gamma, z, y) \quad \forall \gamma \notin x \cup z \cup y \qquad (4.e)$$

The axioms in (4) are clearly satisfied for vertex separation in graphs. (4.e) is the counter-positive form of connectedness transitivity, stating that, if x is connected to γ and γ is connected to y, then x must also be connected to y. (4.d) states that, if z is a vertex cutset separating x from y, then removing additional vertices w from the graph still leaves x and y separated. (4.c) claims that, if x is separated from w with y removed and, simultaneously, x is separated from y with w removed, then x must be separated from both y and w.

The logical independence of the five axioms can be demonstrated by letting U contain four elements and showing that it is always possible to contrive a subset I of triplets (from the subsets of U) which violates one axiom and satisfies the other four. The proof of Theorem 1 [Pearl and Paz 1985] also provides a simple method of constructing the unique graph G, which is isomorphic to I -- starting with a complete graph, we simply delete every edge (α, β) for which a triplet of the form (α, ζ, β) appears in I.

Having a complete characterization for vertex separation in graphs makes it easy to test whether a given model of dependency lends itself to graphical representation. In fact, it is easy to show that the unrestricted intuitive notion of informational relevancy will, in some context, violate each of the last three axioms. Axiom (4.d) is clearly violated in the non-monotonic example of the preceding subsection. Transitivity (4.e) is violated by that same example because reading difficulties may depend on both the paper quality and the ambient light; yet the latter two are independent of each other. (4.c) is violated in contexts where the propositions y and w logically constrain one another. For instance, if y stands for the proposition "The water temperature is above freezing," and w stands for "The water temperature is above 32^oF," then, clearly, knowing the truth of either one of them renders the other superfluous. Yet, contrary to (4.c), this should not render both y and w irrelevant to a third proposition x, say, whether we will enjoy swimming in that water.

Having failed to provide isomorphic graphical representations for even the most elementary models of informational dependency, we settle for the following compromise: Instead of insisting on complete graph isomorphism, we will consider I-maps which may not be D-maps. However, succumbing to the fact that some independencies may escape representation, we will insist that their number be kept at a minimum or, in other words, that the graphs in those maps should contain no superfluous edges.

3. DEPENDENCY MODELS WITH MINIMAL I-MAPS

3.1 Formal Characterization

Definition: A graph G is a *minimal* I-map of dependency model M if no edge of G can be deleted without destroying its I-mapness.

We now define a class of dependency models which possess unique, easily constructed minimal I-maps.

Definition: A *graphoid* is a set I of triplets (x, z, y) where x, z, y are three non-intersecting subsets of elements drawn from a finite collection $U = \{\alpha, \beta, \cdots\}$, having the following four properties. (We shall write $I(x, y, z)$ to state that the triplet (x, y, z) belongs to graphoid I.)

Symmetry

$$I(x, z, y) <\!\!=\!\!> I(y, z, x) \qquad (5.a)$$

Subset Closure

$$I(x, z, y \cup w) \Longrightarrow I(x, z, y) \;\&\; (x, z, w) \qquad (5.b)$$

Intersection

$$I(x, z \cup w, y) \;\&\; I(x, z \cup y, w) \Longrightarrow I(x, z, y \cup w) \qquad (5.c)$$

Union

$$I(x, z, y \cup w) \Longrightarrow I(x, z \cup w, y) \qquad (5.d)$$

Obviously, every graph-isomorphic dependency is a graphoid, but not vice-versa. The first three properties in (5) are identical to those in (4), while the transitivity requirement (4.e) is waived. Moreover, the union property (5.d) is weaker that (4.d) in that it severely restricts the conditions under which a cutset z can be enlarged by w. In the context of informational dependency, this restriction amounts to saying that learning new facts w will not help an irrelevant fact (y) become relevant if the learned facts (w) were, themselves, irrelevant to begin with.

Theorem 2: Every graphoid I has a unique edge-minimum I-map, $G_0 = (U, E_0)$, constructed by connecting *only* pairs (α, β) for which the triplet $(\alpha, U-\alpha-\beta, \beta)$ is not in I, i.e.,

$$(\alpha, \beta) \notin E_0 \quad \textit{iff } I(\alpha, U-\alpha-\beta, \beta) \qquad (6)$$

Definition: A *relevance sphere* $R_I(\alpha)$ of an element $\alpha \in U$ is any subset S of elements for which

$$I(\alpha, S, U-S-\alpha) \text{ and } \alpha \notin S \qquad (7)$$

Let $R_I^*(\alpha)$ stand for the set of all relevance spheres of α. A set is called a *relevance boundary* of α, denoted $B_I(\alpha)$, if it is in $R_I^*(\alpha)$ and if, in addition, none of its proper subsets is in $R_I^*(\alpha)$.

$B_I(\alpha)$ is to be interpreted as the smallest set that "shields" α from the influence of all other elements. Note that $R_I^*(\alpha)$ is non-empty because $I(x, z, \varnothing)$ guarantees that the set $S = U-\alpha$ satisfies (7).

Theorem 3: Every element $\alpha \in U$ in a graphoid I has a unique *relevance boundary* $B_I(\alpha)$. $B_I(\alpha)$ coincides with the set of vertices $B_{G_0}(\alpha)$ adjacent to α in the minimal graph G_0.

Corollary 1: The set of relevance boundaries $B_I(\alpha)$ forms a *neighbor system,* i.e., a collection $B_I^* = \{B_I(\alpha) : \alpha \in U\}$ of subsets of U such that

(i) $\alpha \notin B_I(\alpha)$, and

(ii) $\alpha \in B_I(\beta)$ iff $\beta \in B_I(\alpha)$, $\alpha, \beta \in U$

Corollary 2: The edge-minimum I-map G_0 can be constructed by connecting each α to all members of its relevance boundary $B_I(\alpha)$.

The usefulness of Corollary 2 lies in the fact that in many cases it is the relevance boundaries $B_I(\alpha)$ that define the organizational structure of human memory. People find it natural to identify the immediate consequences and/or justifications of each action or event, and

these relationships constitute the neighborhood semantics for inference nets used in expert systems [Duda et al. 1976]. The fact that $B_I(\alpha)$ coincides with $B_{G_o}(\alpha)$ guarantees that many independencies can be validated by tests for graph separation at the knowledge level itself (Pearl, 1985).

3.2 An Illustration: To illustrate the role of these axioms consider a simple graphoid defined on a set of four integers $U = \{(1, 2, 3, 4\}$. Let I be the set of twelve triplets listed below:

$$\begin{aligned} I \quad = \quad & \{(1, 2, 3), (1, 3, 4), (2, 3, 4), \\ & (\{1, 2\}, 3, 4), (1, \{2, 3\}, 4), \\ & (2, \{1, 3\}, 4), + \text{ symmetrical images}\} \end{aligned}$$

It is easy to see that I satisfies (5.a)-(5.d) and thus it has a unique minimal I-map G_0, shown in Figure 1. This graph can be constructed either by deleting the edges (1, 4) and (2, 4) from the complete graph or by computing from I the relevance boundary of each element, i.e., $B_I(1) = \{2, 3\}$, $B_I(2) = \{1, 3\}$, $B_I(3) = \{1, 2, 4\}$, $B_I(4) = \{3\}$.

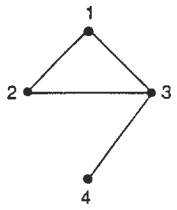


Figure 1: The Minimal I-Map, G_o, of I

Suppose that the list contained only the last two triplets (and their symmetrical images):

$$I' = \{(1, \{2, 3\}, 4), (2, \{1, 3\}, 4) + \text{symmetrical images}\}$$

I' is clearly not a graphoid because the absence of the triplets (1, 3, 4) and (2, 3, 4) violates the intersection axiom (5.c). Indeed, if we try to construct G_0 by the usual criterion of edge deletion, the graph in Figure 1 ensues, but it is no longer an I-map of I'; it shows 3 separating 1 from 4 while (1, 3, 4) is not in I'. In fact, the only I-maps of I' are the three graphs in Figure 2, and the edge-minimum graph is clearly not unique.

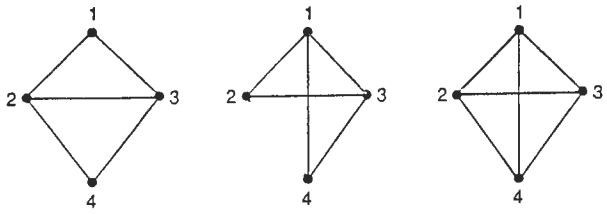


Figure 2: The Three I-Maps of I'

Now consider the list

$$I'' = \{(1, 2, 3), (1, 3, 4), (2, 3, 4), (\{1, 2\}, 3, 4), + \text{images}\}$$

I'' satisfies the first three axioms (5.a)-(5.c) but not the union axiom (5.d). Since no triplet of the form $(\alpha, U-\alpha-\beta, \beta)$ appears in I'', the only I-map for this list is the complete graph. However, the relevance boundaries of I'' do not form a neighbor set; e.g., $B_{I''}(4)=3$, $B_{I''}(2)=\{1, 3, 4\}$, so $2 \notin B_{I''}(4)$ while $4 \in B_{I''}(2)$.

3.3

An Example: Probabilistic Dependencies and Their Graphical Representation

Let $U = \{\alpha, \beta, \cdots\}$ be a finite set of discrete-valued random variables characterized by a joint probability function $P(\cdot)$, and let x, y, and z stand for any three subsets of variables in U. We say that x and y are conditionally independent given z if

$$P(x, y \mid z) = P(x \mid z)\, P(y \mid z) \text{ when } P(z) > 0 \qquad (8)$$

Eq.(8) is a terse notation for the assertion that for any instantiation z_k of the variables in z and for any instantiation x_i and y_j of x and y, we have

$$P(x = x_i \;\&\; y = y_j \mid z = z_k) = P(x = x_i \mid z = z_k)\, P(y = y_j \mid z = z_k) \qquad (9)$$

The requirement $P(z) > 0$ guarantees that all the conditional probabilities are well defined, and we shall henceforth assume that $P > 0$ for any instantiation of the variables in U. This rules out logical and functional dependencies among the variables a case which would require special treatment.

We shall use the notation $(x \perp z \perp y)_P$ or simply $(x \perp z \perp y)$ to denote the independence of x and y given z. Thus,

$$(x \perp z \perp y)_P \text{ if } P(x, y \mid z) = P(x \mid z) P(y \mid z) \qquad (10)$$

Note that $(x \perp z \perp y)$ implies the conditional independence of all pairs of variables $\alpha \in x$ and $\beta \in y$, but the converse is not necessarily true.

The relation $(x \perp z \perp y)$ satisfies the following properties [Lauritzen 1982]:

$$(x \perp z \perp y) \Longleftrightarrow P(x \mid y, z) = P(x \mid z) \qquad (11.a)$$

$$(x \perp z \perp y) \Longleftrightarrow P(x, z \mid y) = P(x \mid z)\, P(z \mid y) \qquad (11.b)$$

$$(x \perp z \perp y) \Longleftrightarrow \exists f, g : P(x, y, z) = f(x, z)\, g(y, z) \qquad (11.c)$$

$$(x \perp z \perp y) \Longleftrightarrow P(x, y, z) = P(x \mid z)\, P(y, z) \qquad (11.d)$$

$$(x \perp z \perp y) \Longrightarrow (x \perp z, f(y) \perp y) \qquad (12.a)$$

$$(x \perp z \perp y) \Longrightarrow (f(x, z) \perp z \perp y) \qquad (12.b)$$

These properties are based on the numeric representation of P and, therefore, would not be adequate for characterizing its graphical representation.

We now ask what *logical* conditions, void of any reference to numerical forms, should constrain the relationship $(x \perp z \perp y)$. The next set of properties constitute such a logical basis.

Theorem 4: Let x, y, and z be three non-intersecting subsets of variables from U, and let $(x \perp z \perp y)$ stand for the relation "x is independent of y, given z" in some probabilistic model P. The following five independent conditions must then hold:

Symmetry

$$(x \perp z \perp y) <==> (y \perp z \perp x) \quad (13.a)$$

Closure for subsets:

$$(x \perp z \perp y, w) ==> (x \perp z \perp y) \ \& \ (x \perp z \perp w) \quad (13.b)$$

Weak closure for intersection:

$$(x \perp z, w \perp y) \ \& \ (x \perp z, y \perp w) ==> (x \perp z \perp y, w) \quad (13.c)$$

Weak closure for union:

$$(x \perp z \perp y, w) ==> (x \perp z, w \perp y) \quad (13.d)$$

Contraction:

$$(x \perp z, y \perp w) \ \& \ (x \perp z \perp y) ==> (x \perp z \perp y, w) \quad (13.e)$$

The intuitive interpretation of Eqs. (13.c) through (13.e) follows. (13.c) states that if y does not affect x when w is held constant and if, simultaneously, w does not affect x when y is held constant, then neither w nor y can affect x. (13.d) states that learning an irrelevant fact (w) cannot help another irrelevant fact (y) become relevant. (13.e) can be interpreted to state that if we judge w to be irrelevant (to x) after learning some irrelevant facts y, then w must have been irrelevant before learning y. Together, the expansion and construction properties mean that learning irrelevant facts should not alter the relevance status of other propositions in the system; whatever was relevant remains relevant, and what was irrelevant remains irrelevant.

The proof of Theorem 1 can be derived by elementary means from the definition (8) and from the basic axioms of probability theory. The intersection property is the only one which requires the assumption $P(x)>0$ and will not hold when the variables in U are constrained by logical dependencies. In such a case, Theorem 1 will still retain its validity if we regard each logical constraint as having some small probability ε of being violated and let $\varepsilon \to 0$.

Obviously, probabilistic dependencies form a graphoid and, therefore, possess the graph properties of Theorems 2 and 3. In particular, we have:

Corollary 3: To every probability distribution P, there corresponds a unique minimal I-map $G_o = (U, E_o)$ constructed by the criterion

$$(\alpha, \beta) \notin E_o \ \textit{iff} \ (\alpha \perp U - \alpha - \beta \perp \beta).$$

Equivalently, G_o can be constructed by connecting each variable α to the smallest set S of variables satisfying

$$P(\alpha \mid S) = P(\alpha \mid U - \alpha)$$

4. CONCLUSIONS

We have established an axiomatic characterization of dependency models which are representable by graphs, and we have identified two essential properties: weak closure for intersection (5.c), and weak closure for union (5.d). These two axioms enable us to construct an edge-minimum graph in which every cutset corresponds to a genuine independence condition, and these were chosen, therefore, as the formal definition of graphoid systems — a general model of informational dependency. Vertex separation in graphs, probabilistic independence and partial uncorrelatedness are special cases of graphoid systems where the two defining axioms are augmented with additional requirements.

The graphical properties associated with graphoid systems offer an effective inference mechanism for deducing, in any given state of knowledge, which propositional variables are relevant to each other. If we identify the relevance boundaries associated with each proposition in the system, and treat them as neighborhood relations defining a graph G_0, then we can correctly deduce irrelevance relationships by testing whether the set of currently known propositions constitutes a cutset in G_0.

REFERENCES

Duda, R. O., Hart, P. E., and Nilsson, N. J., (1976), Subjective Bayesian Methods for Rule-Based Inference Systems, *Proceedings, 1976 NCC (AFIPS)*, 45, 1075-1082.

Lauritzen, S.L. (1982), *Lectures on Contingency Tables*, 2nd Ed., U. of Aalborg Press, Aalborg, Denmark.

Montanari, U. (1974), Networks of Constraints, *Information Science*, Vol. 7, pp. 95-132.

Pearl, J. (1985), Fusion, Propagation and Structuring in Bayesian Networks, Technical Report CSD-850022, June 1985.

Pearl, J. & Paz, A., (1985), GRAPHOIDS: a Graph-Based Logic for Reasoning about Relevance Relations, *Technical Report CSD-850038*, December 1985.

Woods, W.A. (1975), "What's in a Link: Foundations for Semantic Networks" in Bobrow and Collins (eds.), *Representation and Understanding*, Academic Press, Inc.

A Proposal of Modal Logic Programming
(Extended Abstract)

Seiki Akama
Fujitsu Ltd.
2-4-19, Sin-Yokohama, Yokohama, 222, Japan.

Abstract
A theory of logic programming based upon modal logic is investigated. The modal system S5 can be interpreted as an extension of Prolog by adding modal operators. A model theory for generalized modal logic is described as the form of programming language.

1. Introduction

Logic programming languages such as Prolog are generally implemented in a Horn clause subset of first-order predicate calculus. Here an efficient computing formalism is obtained by restricting it to the so-called SLD-resolution. Recently several attempts to extend logic programming to full first-order logic or to other non-standard logics are done in order to strengthen its representability as a programming language. We can regard modal logic as a candidate for such an extension. As is well-known S5 modal logic can be regarded as logic for programs, as in Floyd-Hoare logic, dynamic logic, and S4 modal logic is for the theory of knowledge and action. And we fail to account for theories of belief and knowledge in AI within first-order logic. In fact it is important to formalize various modal concepts for knowledge representation. Therefore it is worth investigating modal logic programming from such considerations.

2. Modal Logic and its Model Theory

In natural language a lot of modalities appear but we cannot formulate such concepts in an ordinary first-order logic. We thus need a modal logic, which is the logic of necessity and possibility, in order to accommodate them. The introductory works on modal logic as Hughes and Cresswell[10,11], Chellas[4] will give a complete exposition. We here discuss the model-theoretic approach to modal logics as T, S4, and S5 on the basis of Kripke semantics. For simplicity, we confine ourselves to a normal system of the classical modal predicate calculus by the so-called canonical model in which every normal system is complete.

The language for modal predicate logic is defined in terms of the following symbols.

(1) a countably infinite set of n-place predicate letters P, Q, ...
(2) a countably infinite set of parameters x, y, ...
(3) the six symbols ~, ∨, L, ∀, (, and).

Other logical symbols are defined in terms of the above mentioned symbols. The formation rules are the same as in ordinary predicate logic. In modal logic two modal operators L(necessity) and M(possibility) are added. We take L as a primitive and define M as:

$MA \equiv \sim L \sim A$.

We write $\vdash_S A$ for the theoremhood of A in S and $\models_S A$ for validity in S, and usually omit a subscript as a syntactic suggar. A normal modal predicate system S is a set of well formed formulas (wff) which contains all tautologies of predicate logic, and the following K, BF, the so-called Barcan formula,

K: $L(A \supset B) \supset (LA \supset LB)$,
BF: $\forall x LP(x) \supset L \forall x P(x)$,

closed under modus ponens(MP), necessitation(N) and generalization(V), i.e.,

MP: $\vdash A$ and $\vdash A \supset B \Rightarrow \vdash B$,
N: $\vdash A \Rightarrow \vdash LA$,
∀: $\vdash A(t) \Rightarrow \vdash \forall x A(x)$.

Let S be a class of formulas of the language of the modal logic. Our next task is to define a Kripke model for the modal logic. A Kripke model for S+BF system (simply we call it BF model) is the triple $\langle F, D, V \rangle$ where F is a frame which consists of a non-empty set of possible worlds W and a binary relation R on F, D is a domain of individuals, and V is an evaluation function of the individual variables and the predicate symbols satisfying the following

(1) For any individual variable x, $V(x) \in D$.
(2) For any n-place predicate symbol P, V(P) is a set of n+1-tuples $\langle u1, \ldots, un, w \rangle$ where each of $u1, \ldots, un$ is in D and $w \in W$.

The interpretation of a wff is an evaluation function which assigns an element of D to the individual variables with respect to the world. We use the following notation:

$[F, g, w] \models A$

for the interpretation g satisfies the wff A at the world w in the frame F. The interpretation is recursively defined by induction on the length of formulas.

(1) $[F, g, w] \models P(x1, \ldots, xn)$ iff $\langle V(x1), \ldots, V(xn), w \rangle \in V(P)$.
(2) $[F, g, w] \models \sim A$ iff $[F, g, w] \not\models A$.
(3) $[F, g, w] \models A \wedge B$ iff $[F, g, w] \models A$ and $[F, g, w] \models B$.
(4) $[F, g, w] \models LA$ iff $[F, g, w'] \models A$ for any $w' \in W$ such that wRw'.
(5) $[F, g, w] \models \forall x A(x)$ iff $[F, g, w'] \models A(d)$ for

any $w' \in W$ such that wRw' and any $d \in D$.

We say that wff A is valid in the model iff $[F,g,w] \models A$ for any $w \in W$. And we say that F is a frame for a system S+BF iff every theorem of S+BF is valid in M. The following systems are familiar to us, namely:

K: BF+{L(A ⊃ B) ⊃.LA ⊃ LB},
T: K+{LA ⊃ A},
S4: T+{LA ⊃ LLA},
S5: S4+{~LA ⊃ L~LA}.

Easily we can understand that T is a reflexive frame, S4 a reflexive transitive frame, and S5 an equivalence frame respectively. Next we define a canonical model for S+BF where S is any normal propositional modal system. In a canonical model for S+BF every non-theorem is false in some world. The model for S+BF is a triples <F,D,V> where

(1) F = <W,R> is a frame in which W is the set of all maximal S+BF-consistent wffs with ∀ property and for any w and w'such that wRw'iff {A|LA ∈ w} ∈ w'. Here ∀ property implies for any A and x, if $\Gamma \vdash A(t)$ for any t, then $\Gamma \vdash \forall x A(x)$.
(2) D is the set of all individual variables.
(3) V is an evaluation function such that
  (a) for any individual variable x, V(x) = x,
  (b) for any n-place predicate symbol P, any individual variables x1,...,xn, and any $w \in W$, <x1,...xn,w> ∈ V(P) iff P(x1,...,xn) ∈ w.

We shall present the following fundamental theorem for a canonical model.

Theorem 1.
If S is a normal modal system and <F,D,V> is the canonincal model for S+BF, then for any wff A and any $w \in W$, $M \models A$ iff $A \in w$.

The proof is by induction on the complexity of formulas. Every theorem of S+BF is in every world in the canonical model. As a consequence of the fact, every theorem of S+BF is valid in the canonical model, namely, we get the completeness theorem for S+BF.

Theorem 2.
For any wff A, A is valid in the canonical model for S+BF iff $\models_{S+BF} A$.

The proof of the theorem is in the above mentioned textbooks for details. Also the completeness theorem implies that the system is characterized by a class of its models or frames.

Theorem 3.
If S+BF is complete then it is characterized by the class of all frames for S.

Interestingly there exists a normal modal system which is not characterized by any class of frame. We call such systems incomplete. The topic is beyond the scope of the present paper. Cresswell[6] investigated an incomplete decidable modal logic for details.

3. Theorem Prover for Modal Logic

In this section we propose a theorem prover for S5 as an extension of Robinson's resolution principle[15]. This implies modal extension of logic programming as Prolog.

First we define a modal conjunctive normal form as follows:

A1 ∧ ... ∧ An

where each Ai($1 \leq i \leq n$) is a disjunction of literals or modal literals. Modal literals are either a modal formula or its negation. A modal formula has the form of $\Delta C(t1,\ldots,tn)$ where Δ is a sequence of modal operators and each ti is a term and C is an n-place predicate symbol. For any formula of the modal logic there exists an equivalent formula in modal conjunctive normal form. We call each disjunct in a modal conjunctive normal form a modal clause, and here we restrict it to Horn(definite) clause which is in the form ~A ∨ B, where A is a conjunction of atomic formulas and B is a modal formula. Each clause is properly generalized if it contains free variables.

We shall add the following inference rules to the ordinary resolution principle,

$$(1)\quad \frac{A \vee \square L(C \vee D) \qquad B \vee \square((\sim D \vee E) \wedge F)}{A \vee B \vee \square((C \vee E) \wedge F)}$$

$$(2)\quad \frac{A \vee \square M(C \vee D) \qquad \square(\sim D \vee E) \wedge F)}{A \vee \square M((C \vee E) \wedge F)}$$

where □ is one of the modal operators, as L, M, or the empty symbol.

Given a set S of modal clauses, we define an SLMD-resolution (SL-resolution for Modal Definite Clause) as a finite (or infinite) sequence of modal clauses each of which is either a modal clause or a resolvent of clause and unit clause. More precisely,

The SLMD-derivation of S {A} is a finite (or infinite) sequence of the triple <Gn,dn+1, λn+1 > ($0 \leq n$) where Gn is a goal clause (G0 = A), dn+1 is an input clause and λn+1 is a most general unifier. Here Gn+1 is a resolvent of Gn and dn+1 by way of λn+1. The final goal in the derivation is an empty clause □.

Here we shall exemplify a deduction step of SLMD-resolution. Consider the following set S of clauses.

1. L~A.
2. L(A ∨ B).
3. M((C ∨ ~B) ∧ D).
4. L~C.
5. ~MD.

The deduction from S is carried out as follows.

6. LB. (by 1 and 2)
7. M(~B ∧ D). (by 3 and 4)
8. M~B. (by 5 and 7)
9. □. (by 6 and 8)

We shall prove Robinson's Resolution theorem in our

system.

Theorem 4.

A set S of modal definite clauses for S5 is unsatisfiable iff there is an SLMD-derivation.

Sufficiency is trivial. For necessity, assume S is unsatisfiable, we proved by induction on the number of symbols in each disjunct in the clause in S. If all clauses are unit clauses (namely n = 0), it is clear from the soundness of SLMD-refutation deleting all modal operators in S. Consider the case of n≠1. The inductive steps are divided into two parts.

(case 1) Assume S contains a modal unit clause $A = \Sigma C$. We delete the modal clause from S, and replace modal Horn clause $\Sigma Di \vee Ei (0 \leq i)$ by the disjunction of $Ej (i \preceq j)$ in S. We designate the set of such clauses as S'.

If $\{\Sigma Di \mid i \succeq 0\} \cup \{\Sigma C\}$ is unsatisfiable then S is also unsatisfiable. By inductive hypothesis there is an SLMD-derivation of S'. By the assumption of deleted clauses there exists an SLMD-derivation of S.

(case 2) Assume S contains a binary Horn modal clause $\sim A \vee \Sigma B$. Our strategy is similar to the case 1. Namely, we shall delete the set of the form of $\{\sim Ai \vee \Sigma Di \mid i \succeq 0\}$ from S, and obtain the set S' assuming $\{\Sigma Di \mid i \succeq 0\} \cup \{\Sigma B\}$ is unsatisfiable. By inductive hypothesis there is an SLMD-derivation of S'. By assumption of deleted clauses we can easily construct an SLMD-derivation of B. Thus there is an SLMD-derivation of S. We can prove the existence of SLMD-derivation of the set of Horn clause S by induction on the number of negative literals in the Horn clauses. This completes the proof of theorem 4. Q.E.D.

This therorem can be extended to other modal logics such as T, S4 etc. without any difficulties. We are to present the foundation of modal logic programming based on the extension of SLD-resolution. But it is an open question whether such a basis is really 'computational' as a programming language. We can also provide a finite axiomatization of modal logic programming introducing modal operators as meta-predicate. In such an approach, it is necessary to give a suitable control component in meta-level. But, our proposal enables us to define modal operators directly in object-level. Clear semantics is also available on the basis of Kripke semantics as long as we confine ourselves to fixed modal system. Thus it may be less flexible to broader contribution to the computation in AI. Several attempts to provide a modal logic programming appeared in, for instance, Brown[3], Warren[17].

4. Logic Programming based on Generalized Modal Logics

Traditional modal logics seem to be insufficient for the application to 'computations' of programming language. If the accommodation to such requirements is possible within logic programming we obtain more powerful tools for AI. Here we shall carry out the extension of modal logics along the way in dynamic logic as in Pratt[14], Harel[9]. We will call such logics generalized modal logics (GML). The model theory for GML is almost the same as Kripke model for modal logics described earlier. We regard each state as a possible world and accesibility as some relation at a possible world. A Kripke model $\langle W, R_G, D, V\rangle$ for GML is essentially the same as the one in the previous section except the following conditions.

$w \models L_G A$ iff $w' \models A$ for any $w' \in W$ such that $wR_G w'$

$w \models M_G A$ iff $w' \models A$ for some $w' \in W$ such that $wR_G w'$

where L and M are generalized modal operators which correspond to [a]P and <a>P respectively in Pratt's notation. Namely, the former means that after the execution of 'a ∈ G', P holds, and the latter after the execution of 'a' it is possible to hold P where G denotes the set of statements.

Suppose the situation where we want to describe the temporal specification (for events). In temporal logics if an atomic sentence A has been held at all the time in the past, we designate it as HA. And we designate by PA when A was true at some past time. Such operators are defined in temporal logic as follows:

$t \models PA$ iff $t' \models A$ for some t' such that tRt',

$t \models HA$ iff $t' \models A$ for any t' such that tRt'.

Here R denotes the temporal precedence relation, and t, r, s elements of W. Restricting a certain system to the basis of modal logic programming, such requirements will cause a language to revise a mechanism of computation. In modal logic programming the above examples are represnted as:

$t \models PA$ iff $t' \models A$ for some t' such that $tR_G t'$,

$t \models HA$ iff $t' \models A$ for any t' such that $tR_G t'$,

where R_G denotes a reflexive transitive relation (corresponding to S5) on moments of time. These are the restrictions to the relation of ordinary temporal logic, namely,

$\forall t(tRt)$,

$\forall t \forall s \forall r(tRs \wedge sRr \supset tRr)$,

hold. But such restrictions will be varied. For example, if we wish to accommodate the concept of branching time as in Ben-Ari et al.[1], the following constraint on R is in order:

$\forall t \forall s \forall r(tRr \wedge sRs \supset tRs \vee t = s \vee sRt)$.

This constraint enables us to add the next axiom to the system, namely,

$PA \wedge PB \supset (P(A \wedge B) \vee P(A \wedge PB) \vee P(PA \wedge B))$.

If the relation is changed, it is often the case that the computation rules cannot continue to be fixed in a programming language.

It is to be noticed that the interpretation of predicate and function symbols in our logic is changeable, if needed, in each possible world. We here adopt the fixed interpretation of them for simplicity. An alternative way to overcome such obstacles is to introduce a channel variable. It is left for further investigations. The model in this

section is the one of tentative formalisms of modal logic programming.

6. Conclusions

Final remarks concern several topics related to AI not investigated in the previous sections related to modal logic programming. First it can be a tool for natural language understanding systems. As the development of the so-called Montague grammar(MG) and logic grammar such as definite clause grammar(DCG), logic programming has made use of natural language analysis as a parser or a question answering system. The interested reader can consult several works as Dowty et al.[7], Pereira and Warren[13], Colmerauer[5] for details. In natural language semantics, modalities play an important role for interpretation of its meanings. This aspect is formalized by means of modal (or intensional) logics as Montague did. Nevertheless Montague's attempts have never become a decisive formalism. In our theory, intensional concepts can be represented by extending model-theoretic structure since we have a Kripke semantics for it. But there exists no finite axiomatization for intensional logic as you know. We think such concepts must be introduced as the meta-level. Only computational formalisms are of use for the application. The extension of MG was carried out in the efficient manners using simpler devices, for example first-order logic, as in Schubert and Pelletier[16]. Another trends aim to weaken the strict compositionality of Frege's principle from the computational and linguistical observations as in Generalized Phrase Structure Grammar(GPSG) developed by Gazdar et al.[8].

Most data base systems require the counterpart of modal logics coping with data base updating, truth maintenances, knowledge acquisition etc. Traditional logic programming systems have no mechanism in order to accommodate them. In data base management meta-level extensions by means of demo predicate in Bowen and Kowalski[2] are used in general. We are now working in formalizing provability logic for logic programming on the basis of GML simulating meta-level inferences in a logic programmming language.

Another topic is a non-monotonic logic. Unlike traditional logic, this logic invalidates old theorems by the introduction of new axioms. McDermott's formalism[12] for non-monotonic logic was presented by modal logics S4, S5, T. Since accessibilities on possible worlds, namely data bases, never continue to be fixed in practical knowledge systems, a model theory for GML must be powerful enough to treat various non-monotonic reasonings.

We hope that a modal logic programming provides a useful tool for AI as decscribed above. We shall present a more elegant formalism and application of GML in a forthcoming paper.

References

[1] Ben-Ari,M.,Manna,Z. and Pnueli,A.,The Temporal Logic of Branching Times, Proc. of 8th ACM Symposium on Principles of Programming Languages, (1981),pp164-176.

[2] Bowen,K.A. and Kowalski,R.A., Amalgamating Language and Metalanguage in Logic Programming, School of Computer and Information Science, University of Syracuse,(1981).

[3] Brown,F.M., A Semantic Theory for Logic Programming, Proc. of Mathematical Logic in Computer Science(1978),pp195-214.

[4] Chellas,B.F., Modal Logic: An Introduction, Cambridge University Press, Cambridge,(1980).

[5] Colmerauer,A., Les Grammaire de Metamorphose, Groupe d'Intelligence Artificielle,Universite de Marseille II,(1975).

[6] Cresswell,M.J., An Incomplete Decidable Modal Logic, JSL 49(1984),pp520-526.

[7]Dowty,D.R., Wall,R.E. and Peters,S.,Introduction to Montague Semantics, Reidel,Dordrecht,(1981).

[8] Gazdar,G.,Klein,E.,Pullum,G.K. and Sag,I.A., Generalized Phrase Structure Grammar, Harvard University Press,(1985).

[9] Harel,D., First-Order Dynamic Logic, Lecture Notes in Computer Science 68, Springer-Verlag, (1979).

[10] Hughes,G.E. and Cresswell,M.J.,An Introduction to Modal Logic, Muthuen, London,(1968).

[11] Hughes,G.E. and Cresswell,M.J., A Companion to Modal Logic, Muthuen, London,(1984).

[12] McDermott,D., Non-monotonic Logic II: non-monotonic modal theories,JACM 29,1(1982), pp33-57.

[13] Pereira,F.C.N. and Warren,D.H.D., Definite Clause Grammar for Language Analysis, Artificial Intelligence 13(1980),pp231-278.

[14] Pratt,V.R., Semantical Consideration on Floyd-Hoare Logic, Proc. of IEEE Symposium on Foundation of Computer Science(1976), pp109-121.

[15] Robinson,J.A., A Machine-Oriented Logic based on Resolution Principle,JACM 12,1(1965),pp23-41.

[16] Schubert,L.K. and Pelletier,F.J., From English to Logic: Context-Free Computation of 'Conventional' Logical Translation,AJCL 8(1982), pp26-44.

[17] Warren,D.S, Database Updates in Pure Prolog, Proc. of Fifth Generation Computer Systems 1984 (1984),pp244-253.

Classical Equality and Prolog

E. W. Elcock and P. Hoddinott
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7

Abstract

The paper considers and formalizes an intuitively appealing notion of extended unification proposed by Kornfeld. The authors show that the formalism supports augmenting Prolog to allow a useful partial implementation of classical equality in selected domains of applications.

Resume

L'article considère et formule une idée intuitivement attirante de l'unification étendue proposé par Kornfeld. Les auteurs montrent que la formalism est an faveur de Prolog étendu pour permèttre une exécution partielle, qui est profitable, de l'égalité classique dans les domains choisis d'application.

1 Introduction

The work by van Emden and Lloyd [13] shows how the notion of a *general procedure* augmented by a particular *equality theory* can be used to make overt the logical framework common to apparently quite different systems. In their paper they present an account of Prolog II [13], as essentially the general procedure together with an appropriate equality theory. As they remark, this is particularly interesting in that Colmerauer's own presentation of Prolog II is one in which Prolog II is regarded as a system for manipulating infinite trees, and presented as a complete departure from a system based on first order logic.

The paper by van Emden and Lloyd makes reference to other novel work which attempts to incorporate equality into Prolog programming. In particular they refer to the work by Kornfeld [8].

Kornfeld's work in this area is particularly provocative. As Goguen [5] points out, Kornfeld gives no theoretical justification for his approach, and that it is in fact incomplete - although Goguen gives no clarification of his criticism of incompleteness that would not apply to all feasible logic programming implementations. Nevertheless, the underlying notions of Kornfeld's work are intuitively appealing. In what follows, we will use the method of van Emden and Lloyd to formalize, analyse and extend Kornfeld's work.

Section 2 introduces Kornfeld's work, and section 3 considers its relationship to classical equality. Sections 4 and 5 discuss theoretic and pragmatic aspects of it within the framework of the homogeneous form and an equality theory. We will be particularly concerned solely with the issues, largely neglected by Kornfeld himself, which bear on the standard equality theory of first order logic.

2 Kornfeld's Implementation of an Equality for Prolog

Kornfeld modified a Lisp embedded Prolog [7] to make an intuitively appealing change in the behaviour of the interpreter on unification failure. Kornfeld's informal description of the change can be paraphrased as follows: If the interpreter attempts to unify two terms Φ and Ψ and fails, then the interpreter attempts to establish the goal $equals(\Phi,\Psi)$. If this goal succeeds, the resulting bindings are added to the binding environment and the original unification is deemed to have succeeded. If the goal fails then the goal $equals(\Psi,\Phi)$ is tried. If this succeeds then the resulting bindings are added to the binding environment and the original unification is deemed to have succeeded: otherwise the original unification is deemed to have failed. We will call this mechanism *extended unification* (e-unification).

This informal description leaves much to be desired in the way of specificity: much of the operational semantics has to be induced from Kornfeld's examples. We will begin by using some of Kornfeld's examples to motivate our interpretation of what we perceive as the intended operational semantics of e-unification.

Consider the introduction of the concept of rationals as a quotient set of ordered pairs of integers. To do this we have to define an equivalence relation over ordered pairs. This equivalence relation can then be used to define equality over the rationals. Kornfeld does this by introducing the axiom

$$E1rat: \quad equals(rat(XN,XD),rat(YN,YD)) \leftarrow times(XN,YD,Z), times(XD,YN,Z).$$

Consider now the first of Kornfeld's examples illustrating e-unification. The terms $rat(2,3)$ and $rat(X,6)$ are said to be e-unifiable. We are told that (standard) unification is attempted first and fails. Kornfeld tells us that this failure results in e-unification generating $equals(rat(2,3), rat(X,6))$, and that this succeeds by $E1rat$, binding X to 4.

Let us now consider this example in more detail. Standard unification would attempt to unify $rat(2,3)$ with $rat(X,6)$ as follows: starting with the leftmost symbol of each term, the algorithm would find a disagreement at the second symbol and would attempt to unify 2 with X. This would succeed, binding X to 2. The next disagreement would lead to an attempt to unify 3 with 6, which would fail. Kornfeld's explanation does not give the impression that e-unification would now lead to a call of $equals(3,6)$, that is, a call to "equals" as a result of (standard) unification failure for a subterm.

We therefore assume that the e-unification of a goal $p(t_1, \ldots, t_n)$ with the head $p(s_1, \ldots, s_n)$ of a clause is successful *iff* for each i, $i \leq n$, the argument t_i is e-unifiable with the argument s_i. The e-unification of an argument t_i with an argument s_i is successful ***if*** t_i and s_i are unifiable, ***else if*** $equals(t_i,s_i)$ can be established, ***else if*** $equals(s_i,t_i)$ can be established.

Let us now consider Kornfeld's formalization of the concept that a rational is equal to an integer. Kornfeld formalizes the equality with the following non-logical axioms:

$E2rat$: $equals(rat(N,D),I) \leftarrow integer(I), var(N), var(D), N = I, D = 1.$

$E3rat$: $equals(rat(N,D),I) \leftarrow integer(I), times(D,I,N).$

The predication $integer(I)$ is a Prolog [1] evaluable predicate which is true just in case I is an integer, and fails otherwise. The predication $var(X)$ is a Prolog evaluable predicate which is true just in case X is an variable, and fails otherwise. The predication $t = s$ is a Prolog evaluable predicate which is now true just in case t and s are e-unifiable, and fails otherwise.

The following is Kornfeld's second example illustrating what we have called "e-unification". Kornfeld says that in the presence of the axioms $E1rat$, $E2rat$ and $E3rat$, the goal

$$\leftarrow mem(rat(4,X), [2,3,cons(Y,Z),rat(R,W),rat(2,7)])$$

will succeed three times with respectively, X bound to 2, R bound to 4 and X bound to W, and X bound to 14.

Again let us consider this example in more detail. It suffices to simplify the example so that the goal, G, is $\leftarrow mem(rat(4,X),[2])$. As Kornfeld does not give an axiomatization of "*mem*", we assume "*mem*" has the following standard Prolog axiomatization:

$Mem1$: $mem(A,[A|L]) \leftarrow$
$Mem2$: $mem(A,[B|L]) \leftarrow mem(A,L).$

If the goal G is to succeed it must be e-unifiable with axiom $Mem1$.

The e-unification of the goal predicate proceeds as follows: The e-unifier first attempts to unify $rat(4,X)$ with A. The unification of $rat(4,X)$ with A succeeds with $rat(4,X)$ bound to A. Next, an attempt is made to unify $[2]$ with $[rat(4,X)|L]$. This attempt fails, and by our earlier assumption the e-unifier now generates $\leftarrow equals([2],[rat(4,X)|L])$. By inspection of axioms, $E1rat$, $E2rat$ and $E3rat$, it is clear that this and the symmetrized version $\leftarrow equals([rat(4,X)|L],[2])$ both fail. Accordingly, G fails, apparently contradicting Kornfeld's claim about this example.

We can however remove the contradiction. Although Kornfeld nowhere discusses their presence or necessity in his system, problems like that above do not arise if we assume that, for every function in the lexicon of the program, we introduce a substitutivity axiom. In the example above, let us add the axiom

$E1list$: $equals([H1|T1],[H2|T2]) \leftarrow H1 = H2, T1 = T2.$

It is easy to see that the goal $\leftarrow equals([2],[rat(4,X)|L])$ which failed in the previous analysis, now succeeds.

In what follows we will take the view that e-unification is formalized as assumed but that where appropriate, equality theories that build on e-unification will include appropriate function substitutivity axioms.

3 Extended unification and classical equality

In this and the following sections we will exploit the possibility of using e-unification to augment Prolog by full classical equality - a possibility not discussed by Kornfeld himself, whose concern would seem to be with the potential of e-unification as a practical mechanism and the "special effects" that can be attained by the default to the distinguished predicate "*equals*". Elcock and Hoddinott [3] considers obtaining these effects using the homogeneous form of a program.

Although the main thrust of our treatment of equality using e-unification is, like the examples above, aimed at a Prolog program with the goal evaluated using a standard Prolog interpreter, we will find it useful and interesting in this section to consider a non-deterministic generalization of e-unification, **-unification*. This allows us to address Goguen's position [5] referenced in the introduction, and give a semantics for e-unification which indicates its relation to classical equality.

The set $\mathcal{E}$ of axioms defining the logical characteristics of classical equality [11] are:

$\mathcal{E}1$: $equals(X,X) \leftarrow$

$\mathcal{E}2$: $equals(X,Y) \leftarrow equals(Y,X)$

$\mathcal{E}3$: $equals(X,Y) \leftarrow equals(X,Z), equals(Z,Y)$

$\mathcal{E}4$: For all predicates p:

$p(X_1,\ldots,X_n) \leftarrow p(Y_1,\ldots,Y_n), equals(X_1,Y_1),\ldots,equals(X_n,Y_n)$

$\mathcal{E}5$: For all functors f:

$equals(f(X_1,\ldots,X_n), f(Y_1,\ldots,Y_n)) \leftarrow equals(X_1,Y_1),\ldots,equals(X_n,Y_n)$

(i.e. reflexivity, symmetry, transitivity, predicate replaceability and function substitutivity, respectively).

We define the **-unification** of a term t with a term s as the non-deterministic selection of one of the following operations: (1) unifying t and s, (2) generating $\leftarrow equals(t,s)$, or (3) generating $\leftarrow equals(s,t)$. Let us now consider this generalization of e-unification and its relationship to classical equality. In this paper our treatment of the relationship of **-unification** to classical equality is informal. A formal treatment is given in [4] which also proves the soundness and completeness of SLD-resolution [9, 12] augmented by **-unification*.

First, let us consider the predicate replaceability axioms: Suppose the goal $\leftarrow p(t)$ is resolved with the predicate replacement axiom $p(X) \leftarrow p(Y), equals(X,Y)$ to obtain $\leftarrow p(Y), equals(t,Y)$ and that the goal $p(Y)$ in this goal clause is then resolved with $p(s) \leftarrow B$ to obtain $\leftarrow equals(t,s), B$. This last goal clause can also be obtained by resolving $\leftarrow p(t)$ with $p(s) \leftarrow B$ using the operation of generating $\leftarrow equals(t,s)$ from the **-unification** of $p(t)$ and $p(s)$.

Secondly, let us consider the transitivity axiom: Suppose the goal $\leftarrow equals(t_1,t_2)$ is resolved with the transitivity axiom $equals(X,Y) \leftarrow equals(X,Z), equals(Z,Y)$ to obtain $\leftarrow equals(t_1,Z), equals(Z,t_2)$ and that the goal $equals(t_1,Z)$ in this goal clause is then resolved with $equals(s_1,s_2) \leftarrow B$ to obtain $\leftarrow (equals(s_2,t_2), B)\sigma$, where σ is the most general unifier of t_1 and s_1. This last goal clause may be obtained with resolution using **-unification** by resolving $\leftarrow equals(t_1,t_2)$ with $equals(s_1,s_2) \leftarrow B$ as follows: the **-unification** of t_1 and s_1 unifies these terms having most general unifier σ, and the **-unification** of t_2 with s_2 generates the goal $\leftarrow equals(s_2,t_2)$.

Lastly, let us consider the symmetry axiom: Suppose the goal $\leftarrow equals(t,s)$ is resolved with the symmetry axiom $equals(X,Y) \leftarrow equals(Y,X)$ to obtain $\leftarrow equals(s,t)$. This goal may be obtained with resolution using **-unification** by resolving $\leftarrow equals(t,s)$ with $equals(X,X) \leftarrow$ as follows: the **-unification** of t and X generates the goal $\leftarrow equals(X,t)$, and the **-unification** of s with X unifies these terms.

The significance, for this section, of recasting Kornfeld's e-unification in terms of the non-deterministic generalization, **-unification**, is that it reveals e-unification to be an implementation of a mechanism, **-unification**, which captures the axioms $\{\mathcal{E}2, \mathcal{E}3, \mathcal{E}4\}$ of classical equality. It should be noted that **-unification** of itself does not capture $\mathcal{E}1$ and $\mathcal{E}5$.

4 The homogeneous form and *-unification.

In this section we show how **-unification** can be captured using the homogeneous form of a program together with an appropriate equality theory. As we are interested in **-unification** and its relationship to classical equality, we consider only those programs P containing $\mathcal{E}1$ and $\mathcal{E}5$ and not containing $\mathcal{E}2$, $\mathcal{E}3$ and $\mathcal{E}4$. For such programs, as discussed in the previous section, **-unification** ensures that the logical characteristics of equality are captured. Note that P may contain one or more domain specific equality axioms such as $E1rat$ presented in section 2.

The *homogeneous form* [13] of a definite clause having the form $p(t_1,\ldots,t_n) \leftarrow B$ is the definite clause having the form

$p(x_1,\ldots,x_n) \leftarrow equals(x_1,t_1),\ldots,equals(x_n,t_n), B,$

where $x_1,\ldots,x_n$ are distinct variables not appearing in the original clause. The homogeneous form $\mathcal{H}(D)$ of a set of definite clauses D is the collection of the homogeneous forms of the clauses in D.

In [6] the authors show that for a goal clause G and a set of definite clauses D, $G \cup D \cup \mathcal{E}$ is unsatisfiable iff $G \cup \mathcal{H}(D) \cup \mathcal{H}(\mathcal{E}5) \cup \{\mathcal{E}1, \mathcal{E}2\}$ is unsatisfiable. In short, the transformation to homogeneous form subsumes $\mathcal{E}3$ and $\mathcal{E}4$. Accordingly, for a program P, the program $\mathcal{H}(P - \mathcal{E}1)$ together with the equality theory $\mathcal{E}1$ and $\mathcal{E}2$ captures **-unification*. More specifically, for a goal clause G and a program P, the set $G \cup P$ is unsatisfiable by SLD-resolution augmented by **-unification** iff $G \cup \mathcal{H}(P - \mathcal{E}1) \cup \{\mathcal{E}1, \mathcal{E}2\}$ is unsatisfiable.

We present an example to illustrate this. Consider the program P:

1: $p(a) \leftarrow$

2: $equals(a,b) \leftarrow$

3: $equals(b,c) \leftarrow$

$\mathcal{E}1$: $equals(X,X) \leftarrow$

A refutation constructed for the goal $\leftarrow p(c)$ using SLD-resolution augmented by **-unification** is displayed as follows:

$\leftarrow equals(a,c)$	by 1
$\leftarrow equals(b,c)$	by 2
$\square$	by 3

The homogeneous form of $P-\mathcal{E}1$ is as follows:

$1'$: $p(X) \leftarrow equals(X,a)$
$2'$: $equals(X,Y) \leftarrow equals(X,a), equals(Y,b)$
$3'$: $equals(X,Y) \leftarrow equals(X,b), equals(Y,c)$

As this program is in homogeneous form it only needs to be augmented by the classical equality axioms

$\mathcal{E}1$: $equals(X,X) \leftarrow$
$\mathcal{E}2$: $equals(X,Y) \leftarrow equals(Y,X)$

to obtain a SLD-refutation for $\leftarrow p(c)$. The refutation is displayed as follows:

$\leftarrow equals(c,a)$	by $1'$
$\leftarrow equals(a,c)$	by $\mathcal{E}2$
$\leftarrow equals(a,a), equals(c,b)$	by $2'$
$\leftarrow equals(b,c)$	by $\mathcal{E}1$ and $\mathcal{E}2$
$\leftarrow equals(b,b), equals(c,c)$	by $3'$
$\square$	by $\mathcal{E}1$

5 Extended unification: pragmatics

In this section we will use the homogeneous form of a Prolog program to sketch the development of a Prolog with equality which itself can be executed using a standard Prolog interpreter such as that for the Dec-10 [10]. The development will be made in a way which is consistent with Kornfeld's extended unification. Indeed, the intent of the exposition is to make clear the potential and the limitations of this mechanism as a practical device.

The discussion of section 4 shows that a non-deterministic version of extended unification can be given a nice characterization using the homogeneous form of a program and an equality theory. The expectation might be that the theoretical framework provided by the homogeneous form and section 4 should underpin an implementation in which, as mentioned, the appropriate equality theory could be clearly expressed as a Prolog procedure for the predicate "*equals*". For the moment we will ignore the well known [2] potential difficulties of clause orderings caused by the search strategy of the standard Prolog interpreter.

First, let us consider implementing the operational semantics of e-unification as a deterministic approximation to **-unification*. Let P and $\mathcal{H}(P-\mathcal{E}1)$ be as before. The program $\mathcal{H}(P-\mathcal{E}1)$ is first transformed as follows: each predication of the form $equals(t,s)$ occurring in the body of a clause is replaced by the predication $\Theta(t,s)$, where Θ is a distinguished predicate not occurring in $\mathcal{H}(P-\mathcal{E}1)$. The intended interpretation of $\Theta(t,s)$ is: t and s are e-unifiable. Once the program is so transformed the following axioms are added:

$E1$: $\Theta(X,X) \leftarrow$
$E2$: $\Theta(X,Y) \leftarrow equals(X,Y)$
$E3$: $\Theta(X,Y) \leftarrow equals(Y,X)$

The textural ordering of these axioms simulates the operational semantics of e-unification. For example, reduction of a goal of the form $\leftarrow \Theta(t,s)$ will, using $E1$, first attempt to unify these terms. If this attempt fails then $E2$ will exchange $\leftarrow \Theta(t,s)$ for $\leftarrow equals(t,s)$. If this goal fails then $E3$ will exchange $\leftarrow \Theta(t,s)$ for $\leftarrow equals(s,t)$.

Let us now consider predicate replaceability: With e-unification, if we have a program containing clauses

1: $p(a) \leftarrow$
2: $equals(b,a) \leftarrow$

then the goal $\leftarrow p(b)$ is reduced by e-unification with 1 to $\leftarrow equals(b,a)$ which succeeds by 2.

In section 4 we mentioned that predicate replaceability was subsumed by the homogeneous form. This carries over to the implementation. Thus, the homogeneous form of $p(a)$ is

$1'$: $p(X) \leftarrow \Theta(X,a)$.

The equality theory contains the clauses

$E1$: $\Theta(X,X) \leftarrow$
$E2$: $\Theta(X,Y) \leftarrow equals(X,Y)$
$E3$: $\Theta(X,Y) \leftarrow equals(Y,X)$
$2'$: $equals(X,Y) \leftarrow \Theta(X,b), \Theta(Y,a)$

The goal $\leftarrow p(b)$ is reduced to $\leftarrow \Theta(b,a)$ by $1'$ which is reduced to $\leftarrow equals(b,a)$ by $E2$, which succeeds by $2'$ and $E1$. Incidentally, the "more direct goal" $\leftarrow p(a)$ is reduced by $1'$ to $\leftarrow \Theta(a,a)$ which succeeds by $E1$.

Transitivity however is unpleasant. Consider the clauses

1: $equals(a,b) \leftarrow$
2: $equals(b,c) \leftarrow$

and the goal $\leftarrow equals(c,a)$. With e-unification this goal reduces by 1 to $\leftarrow equals(c,a), equals(a,b)$ and we have initiated an infinite computation: i.e. transitivity is not captured by e-unification.

If we now consider a deterministic implementation using the homogeneous form we see that because of the homogeneous form of those clauses defining "*equals*" it is very easy to initiate an infinite computation also. Accordingly, we need to supply a procedurally stable implementation of the transitivity axiom. The results in [6] show that it is the conversion of those clauses defining "*equals*" that subsumes transitivity. Because of this and the fact that we are to provide an implementation of transitivity we will, in what follows, not write those clauses defining "*equals*" in homogeneous form.

We will attempt to motivate the development of an implementation of transitivity by successive modifications of the "classical" axiom. Let us begin by augmenting our

basic equality theory for e-unification by an axiom which is a simple transformation of the classical transitivity axiom. Our initial equality theory is:

$E1$: $\Theta(X,X) \leftarrow$
$E2$: $\Theta(X,Y) \leftarrow equals(X,Y)$
$E3$: $\Theta(X,Y) \leftarrow equals(Y,X)$
$E4$: $\Theta(X,Y) \leftarrow equals(X,Z), \Theta(Z,Y)$

Consider an example in which the equality theory includes the ground clauses

5: $equals(b,a) \leftarrow$
6: $equals(b,c) \leftarrow$
7: $equals(c,d) \leftarrow$

We will attempt to satisfy the goal

$$\leftarrow mem(d,[a]).$$

Using the homogeneous form for "mem", this reduces to

$$\leftarrow \Theta(d,X1), \Theta([a],[X1|L])$$

When, using $E1$ we have

$$\leftarrow \Theta([a],[d|L]).$$

When, using the substitutivity axiom for lists, we have

$$\leftarrow \Theta(a,d), \Theta([],L).$$

The first literal is now reduced to

$\leftarrow equals(a,d)$ using $E2$) ... fails
$\leftarrow equals(d,a)$ using $E3$) ... fails
$\leftarrow equals(a,Z), \Theta(Z,d)$ using $E4$) ... fails

Because of the Prolog search strategy we clearly need to include a "redundant" symmetrization in the transitivity axiom. We replace axiom $E4$ above by

$E4$: $\Theta(X,Y) \leftarrow equals(X,Z), \Theta(Z,Y)$
$E4'$: $\Theta(X,Y) \leftarrow equals(Z,X), \Theta(Z,Y)$

The goal in the previous example can now be established.

However, consider the example in which the equality theory consists of the axioms $E1$ to $E4'$ together with the ground clauses

5: $equals(a,b) \leftarrow$
6: $equals(b,a) \leftarrow$
7: $equals(b,c) \leftarrow$
8: $equals(c,d) \leftarrow$

We try to establish the goal

$$\leftarrow \Theta(a,d)$$

We obtain the derivation (omitting failure and backtracking)

$\leftarrow equals(a,Z), \Theta(Z,d)$ using $E4$)
$\leftarrow \Theta(b,d)$ using 5)
$\leftarrow equals(b,Z1), \Theta(Z1,d)$ using $E4$)
$\leftarrow \Theta(a,d)$ using 6)

and again we have an infinite computation.

This time it is clear that we need to detect and prevent loops in the "transitivity chain". A straightforward approach is, whenever a transitivity axiom is called, then record the two terms on an equality list. However, a little thought shows that each application of a transitivity axiom need only record one term. We finally propose the equality theory:

$\Theta(X,X) \leftarrow !$
$\Theta(X,Y) \leftarrow equals(X,Y),!$
$\Theta(X,Y) \leftarrow equals(Y,X),!$
$\Theta(X,Y) \leftarrow \Theta 1(X,Y,[]),!$
$\Theta 1(X,Y,\_) \leftarrow equals(X,Y)$
$\Theta 1(X,Y,\_) \leftarrow equals(Y,X)$
$\Theta 1(X,Y,L) \leftarrow equals(X,Z), not(mem(Z,L)), \Theta 1(Z,Y,[X|L])$
$\Theta 1(X,Y,L) \leftarrow equals(Z,X), not(mem(Z,L)), \Theta 1(Z,Y,[X|L])$

The use of the cut (i.e. !) in this theory prevents backtracking from attempting to re-establish a goal of the form $\Theta(\Phi,\Psi)$.

It is left as an exercise to the reader to show that the goal $\leftarrow mem(d,[a])$ of the previous example now succeeds. As far as we are aware (!), at least this level of complexity is necessary to overcome the problems identified earlier.

There remain certain problems of incompleteness when specifying equality between functions. These are similar to those discussed by Kornfeld [8, p.518] and can be partially "solved" by constraining the allowed structure of the arguments of the terms. However, we do not propose to elaborate the implementation further.

6 Conclusion

The paper considers an intuitively appealing notion of extended unification proposed by Kornfeld [8], in which it appears that a useful partial implementation of the classical equality axioms might be possible. The paper shows that, for a non-deterministic machine, extended unification can be generalized to what we have called **-unification*. This in turn can be used in SLD proofs to capture (part of) classical equality. Finally, we show how this last can be nicely characterized by the homogeneous form together with an appropriate equality theory, thereby making quite clear the power of the device of **-unification* as such.

In section 5 we have used the theoretical framework for classical equality using the homogeneous form to motivate a feasible implementation: a notoriously difficult task. Although incomplete, we are of the opinion that the implementation is simple and clear enough to provide a useful and extensible tool in selected domains of application.

7 Acknowledgements

The authors would like to thank Dr. Ed Stabler (Jr), Dr. W. Demopoulos and K. H. Chan for rewarding discussions. The work was carried out partly with the support of NSERC grant number A2445, and partly with the support of IBM Corporation.

References

1. Clocksin, W. F. and Mellish, C. S. *Programming in Prolog.* Springer-Verlag, New York, 1981.

2. Elcock, E. W. The Pragmatics of Prolog: Some Comments. Report 90, Dept. of Computer Science, The University of Western Ontario, London, Ontario, Canada, April, 1983.

3. Elcock, E. W. and Hoddinott, P. Comments on Kornfeld's "Equality for Prolog": e-unification as a mechanism for augmenting the Prolog search strategy. Report 185, Dept. of Computer Science, The University of Western Ontario, London, Ontario, Canada, 1986.

4. Elcock, E. W. and Hoddinott, P. Extended unification and classical equality. Report 186, Dept. of Computer Science, The University of Western Ontario, London, Ontario, Canada, 1986.

5. Goguen, Joseph A. and Meseguer, J. "Equality, Types, Modules, and (Why Not?) Generics for Logic Programming". *The Journal of Logic Programming 1*, 2 (August 1984), 179-209.

6. Hoddinott, P. and Elcock, E. W. "Building-in" classical equality into Prolog. TR 184, Department of Computer Science, The University of Western Ontario, 1985.

7. Kahn, K. Unique Features of LM-Prolog. Unpublished manuscript.

8. Kornfeld, W. A. Equality for Prolog. Proceedings, Seventh International Joint Conference on Artificial Intelligence, 1983, pp. 514-519.

9. Kowalski, R. A. Predicate Logic as Programming Language. Proceedings, Information Processing, Amsterdam, 1974, pp. 570-574.

10. Pereira, L. M., Pereira, F. C. N. and Warren, D. H. D. USER'S GUIDE to DECsystem-10 PROLOG. Dept. of Artificial Intelligence University of Edinburgh, University of Edinburgh, 1978.

11. Shoenfield, J. *Mathematical Logic.* Addison-Wesley, 1967.

12. van Emden, M. H. Programming with Resolution Logic. In *Machine Intelligence 8*, Ellis Horwood, 1977, pp. 266-299.

13. van Emden, M. H. and Lloyd, J. W. "A Logical Reconstruction of Prolog II". *The Journal of Logic Programming 1*, 2 (August 1984), 143-150.

Diagnosis of Non-syntactic Programming Errors in the SCENT Advisor

Gordon I. McCalla

Richard B. Bunt

Janelle J. Harms

Department of Computational Science
University of Saskatchewan
Saskatoon, Saskatchewan, Canada

ABSTRACT

The process of dispensing useful advice to students on their program bugs is one requiring a great deal of intelligence and knowledge. The SCENT (Student Computing Environment) project is concerned with bringing techniques from AI to bear on the building of an intelligent tutoring system to help student programmers develop and debug their programs. The thrust of current SCENT investigations is into the design of the SCENT advisor, which is meant to provide debugging assistance to novice students. A significant component of this work relates to the diagnosis of non-syntactic errors -- errors at the logical or conceptual level. This paper describes the manner by which such diagnosis can be made, and illustrates the operation of the SCENT system by means of detailed examples.

1. Introduction

The intention of the SCENT project is to investigate the issues associated with the design and construction of a programming environment oriented to the teaching of Computer Science. In the near term the project is focussed on providing sophisticated debugging assistance to novice programmers. In short, the project is attempting to automate the role of the conventional programming advisor.

As students write programs several different types of "errors" are made. Their basic lack of familiarity with the language they are learning causes them to commit various syntactic violations. Normally these are detected and "flagged" by the compiler or interpreter they are using with appropriate error messages, and while their progress may be impeded, the student is usually able to determine the (approximate) cause of the error and a corresponding remedy. Several automated systems have attempted to provide additional interpretation and/or guidance here. An interesting approach is that taken by the GENIUS system [McCalla and Murtagh 1985].

More insidious are the "logic" errors. These may be due to a lack of understanding of the language semantics, the selection of an inappropriate solution strategy, or simply a careless action. As often as not, no error message is produced from such errors; where messages do result, they normally stem from a consequence or side effect of the error rather than from the error itself. Such errors have posed problems for automated systems, but some progress has been made, for example in the Programmer's Apprentice [Rich and Shrobe 1978], in LAURA [Adam and Laurent 1980], and in PHENARETE [Wertz 1982]. LAURA is able to spot bugs in FORTRAN programs by comparing the user's solution to an "ideal" solution. The Programmer's Apprentice is designed to help expert LISP programmers debug both their strategies and their code. PHENARETE helps debug code at three levels: surface (lexical and syntactic), deep (semantic and teleological), and conceptual.

A third type of error, that may in fact be manifested in errors of either of the above types, is particularly important in the student context. This type of error arises from a misconception, that is, from a student's failure to understand one or more of the concepts required to solve the task at hand. Errors of this third type have not yet been the subject of extensive study. Some systems, for example the LISP Tutor [Anderson and Reiser 1985], deal with this problem by channeling the student into pre-ordained strategic directions and not allowing him/her to go too far astray. Perhaps the major work that tackles strategic misconceptions head-on is that by Soloway and his colleagues, with MENO-II [Soloway et al. 1981] and later, PROUST [Soloway and Johnson 1985]. The MENO-II system experiments with how to teach programming concepts and also how to recognize student misconceptions in the framework of a PASCAL tutor. PROUST builds on this work by attempting to recognize the solution method (whether valid or invalid) that the student has employed in his/her program.

SCENT's debugging assistance also attempts to go beyond the detection of simple syntax errors, to include the detection of more subtle kinds of errors, involving inappropriate strategies or conceptual difficulties. Unlike many of the other systems developed or proposed, SCENT attempts to exploit information beyond the code itself to help in the debugging process. This includes execution traces, cross-referencing information, structure charts, etc. It also attempts to integrate knowledge about the student in the diagnosis process. PROUST is perhaps the closest in approach. In particular, its emphasis on recognizing the

student's strategy, its dependence on domain-specific and problem-specific knowledge, its multi-level approach, its liberalism in allowing the student to complete his/her program, and its ability to partially match a solution all find sympathetic echoes in SCENT.

This paper begins with a brief overview of the SCENT system architecture. Attention is then focussed on the components responsible for problem diagnosis as detailed examples are discussed. The paper concludes with some general observations and future directions.

2. General Overview of the SCENT System

An architecture has been developed to accomplish the (short term) goals of the SCENT project. This architecture is illustrated in Fig. 1 and is described in detail in [McCalla, Bunt, and Harms 1985]. A pilot system has been designed and implemented to illustrate the effectiveness of this architecture and the feasibility of the approach.

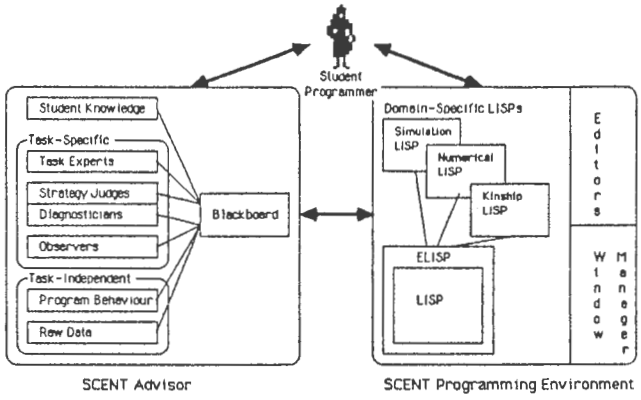


Fig. 1: The SCENT System

Without some constraints, the problem of fully automated error diagnosis would be overwhelming, involving a combinatorially explosive number of possible strategies, implementations, misconceptions, etc. In order to provide a context necessary for any progress, a restricted domain is defined in which are specified a set of selected tasks for the student to solve. The student is free to choose any particular task from this set, and undertakes to write a program to solve this chosen task. The domain chosen for the initial development activity is referred to as the *kinship domain* and is based on having the student write programs to manipulate the familiar family tree.

The kinship domain was selected for several reasons. First, it is familiar, and thus the solution of problems in the domain is not made difficult by lack of understanding of the domain. Even the novice student readily understands problems such as "Find the grandparent of ..." or "Find the siblings of ...". At the same time, the kinship domain is rich in terms of computer science concepts that can be illustrated. Concepts such as procedures, recursion, and tree traversal, for example, are important to the solution of problems in this domain. Other possible domains include a numerical domain, a text-processing domain, and a simulation domain.

As shown in Fig. 1 the SCENT system contains two parts: the programming environment and the advisor. The current SCENT advisor has been designed to analyze students' solutions to problems in the kinship domain, but its methodology is more general than just this domain. In particular, six conceptual levels have been identified. These deal with wide-ranging types of information, ranging from low level program code to quite abstract interpretations of it.

At the lowest level is a student's attempted solution to a particular task, referred to as the *raw data*. At the next level are traces, cross-reference listings, etc., that are derived from straightforward operations applied to this raw data. Patterns observed at this level can indicate important features of the student's solution to the task at hand. The next level of abstraction, therefore, consists of a number of *observers*, each of which is responsible for recognizing a particular pattern. Without further interpretation, these observations would be of limited value. It is the responsibility of *strategy judges* and *diagnosticians* at the next level to bring knowledge of various solution strategies, and possible ways they can go wrong, to the interpretation of these observations. There may be many possible interpretations, and the *task experts* at the next level of abstraction must arbitrate among the various possibilities and choose the interpretation most appropriate to the particular task at hand. At the level of the task expert an interpretation of the student's solution exists, but this interpretation must be related to the student's current goals and level of understanding before appropriate advice can be dispensed. This is the responsibility of the highest level of abstraction, *student knowledge*. Moving up through these levels, interpretations become more task-specific and student-specific. These restrictions are achieved by adding knowledge of the task and student at each level, which, while narrowing the scope at each level, also deepens the interpretation.

It should be noted that while the description of the conceptual levels has proceeded in a bottom-up fashion, this is not meant to imply that information or control actually flows through the various levels in this manner. Rather it is much more heterarchical. In fact, an aggregation of events at a lower level can trigger the activation of a higher level entity to interpret the events, and a higher level entity may require lower-level information to carry out its goals. This suggests that control flow between entities at various levels should be flexible: both top-down and bottom-up invocation strategies should be available; information should be available to entities at any level; it should be possible for many entities to actively consider various requests simultaneously. A control structure is needed that accommodates any of these possibilities, as well as a communication structure that permits entities at various levels to exchange information. This suggests the need for a central information exchange that allows control and data to flow freely. The *blackboard* control scheme, first proposed for the Hearsay II speech understanding system [Erman et al. 1980], provides just such a capability. How the blackboard methodology is adapted for use in SCENT is outlined in [McCalla, Bunt, and Harms 1985].

Initial work on the pilot system has concentrated on providing task experts for several examples in the kinship domain. These task experts have been implemented in LISP and illustrate the range of intelligent debugging capabilities possible in the tasks they address. While they are by no means complete debugging assistants, they have helped to clarify the architecture and suggest ways that new and/or more sophisticated experts can be constructed. Strategy judges, diagnosticians, and observers, along with various traces, cross references listings, etc., have also been devised and have provided useful confirmation of the viability of the SCENT approach. While the student knowledge and blackboard components are currently quite primitive, their role in the system has become much clearer through the initial experiments.

3. The Programming Context

While the main thrust of this paper, and indeed of the entire SCENT project to date, has been in the advisor component, some work has begun in the programming environment component (see Fig. 1). A brief discussion is necessary to provide context for the examples to be presented.

LISP was chosen as the fundamental system language for SCENT because of its symbolic capabilities, its flexibility its interactive nature, its extensibility, and its elegant functional characteristics. All programs in the advisor are written in LISP. Student programs are written in an enhanced version of LISP, not only because of the uniformity this builds into SCENT but also because of LISP's appropriateness as an introductory programming language (see [Abelson, Sussman, and Sussman 1985] for arguments supporting this, perhaps controversial, point of view).

For this application, two levels of LISP extensions have been implemented. These are depicted in Fig. 1. The first level (ELISP) traps LISP error messages for use by the advisor, and also defines several domain-independent extensions to the language that provide useful structuring tools for programs (e.g., an improved **if** construct, a more appropriate procedure definition capability). On top of ELISP are a number of domain-specific LISP enhancements which provide capabilities appropriate to the particular domains. In the kinship domain, for example, are found functions such as **children**, **parent**, etc., as well as a global data structure containing the kinship relationships. Similar extensions would be defined for other domains as well. For example, enhanced numerical capabilities would be required for a domain dealing with numerical computations, and queuing primitives would be essential in a simulation domain. Student programs are written in the appropriate domain-specific LISP, built atop ELISP.

4. The Tasks

In the kinship domain the student writes programs to solve problems such as finding a person's grandparent, cousins, siblings, etc., by searching through a family tree. To facilitate both the expression and the solution of problems in this domain, a specific family tree is provided to the student. This serves several purposes. First, the student need not be concerned with constructing a tree, at least initially. Second, the system can know what the correct answer to a problem is, and also what errors might lead to which wrong answers. The particular tree used for the discussions in this paper is shown in Fig. 2. Each person in the tree has only a single parent, which may be male or female. A number of attributes can be associated with the nodes of the tree, including name and sex. To provide a basis for solutions to the problems posed in this domain, a number of domain-specific primitives are supplied. These include **parent** and **children**, which permit simple navigation about the kinship tree.

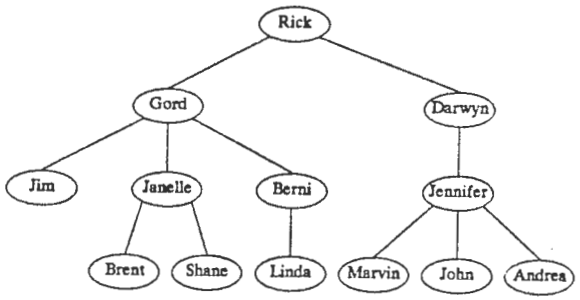


Fig. 2: The Kinship Tree

Problems of varying degrees of difficulty are possible within this domain. Two example problems form the basis for the remaining discussion. The first of these, referred to as *Eldest Ancestor* (EA), requires that the student write a program to determine the eldest ancestor of a given person in the tree. Since all nodes descend from the root, the "correct" answer will always be the name of the root node. The problem, then, is to navigate along the path from the given node to the root by means of successive calls to **parent**, possibly in a recursive manner. While simple, this problem draws upon some important concepts, such as function calling and recursion, and has proven to be a useful testbed for the ideas of SCENT. Because the strategy is relatively simple, the role of the diagnostician is particularly clear-cut here.

The second problem was chosen to be more sophisticated, in the sense of there being several distinct and clearly identifiable strategies that a student might use in his/her solution. Referred to as *Sorted Subtree* (SST), the problem is the following: given a node in the tree (by name), produce a sorted list of all nodes in the subtree with the given node as root. This involves both a traversal of the subtree (using the **children** and **parent** primitives) and a sort. The fact that the SST task has these two identifiable subcomponents marks another point of departure from the simpler EA task: EA is an example of a task that is *non-decomposible*, while SST is an example of a task that is *decomposible*.

5. Sample Diagnoses

In this section, attempts at solution of the two problems introduced in the previous section are discussed. Actual output from the SCENT system prototype is used.

5.1 Diagnosis in the EA Task

The following is a correct solution to the EA problem. Note the use of elements of both ELISP (**defproc**, **if**) and kinship LISP (**parent**).

```
(defproc eldest (x)
  (if (noparent x)
      x
      (eldest (parent x))))
```

The correct answer on the supplied tree (Fig. 2) is "Rick", regardless of what input name is supplied. In this problem most of the errors will result from failure to grasp all of the subtleties of recursion. Among the errors that could be made are moving down the tree instead of up (i.e., calls to **children** as opposed to **parent**), errors in the mechanics of recursion (e.g., the recursive step), or improper temination (e.g., no base case for the recursion). An example of a faulty student solution and the SCENT advisor's analysis of both the behaviour and structure of the student's solution is shown in Example 1.

```
(defproc eldest (x)
  (eldest x))

Ask away:(eldest Rick)
(Your answer is --)(Error in kineval - infinite loop detected)
OUTPUT:(Error in kineval - infinite loop detected)
DIAGNOSTICS: ild ild ild ild  [infinite loop detected]
(checking eldest of Rick:)(Error in kineval - infinite loop detected)
(checking eldest of Gord:)(Error in kineval - infinite loop detected)
(checking eldest of Jennifer:)(Error in kineval - infinite loop detected)
(checking eldest of Linda)(Error in kineval - infinite loop detected)
(You seem to be missing a base case and a reduction step?)
STRUCTURE:(No base case)
```

Example 1

The student supplies his/her attempted solution, and in response to the system prompt,

Ask away:

enters the particular case he/she wishes to run, which is

(eldest Rick)

The system then responds with the outcome of its attempt to run the student's program on the case indicated. In this case the message is a cryptic

(Error in kineval - infinite loop detected)

provided by kinship LISP. This message is typical of the cryptic messages that students receive from conventional environments.

The SCENT advisor, however, is able to offer more as it proceeds to track down the cause of the problem. The Eldest Ancestor Task Expert notes the error and calls in the Eldest Ancestor Diagnostician which begins by running the student's program on four selected test cases (*Rick*, *Gord*, *Jennifer*, and *Linda*). These cases have been selected by virtue of the fact that the output produced by a student's program when run on these inputs is symptomatic of particular errors in the solution. For more sophisticated tasks, the choice of test cases is more complicated and thus requires more thought. Observers monitor the results of these executions, reporting their findings to the blackboard.

From the fact that all of these tests result in the same infinite loop, the Diagnostician in this case deduces that the problem is the lack of a base case and reduction step for the recursion, which causes the search to wander off the tree into Neverland. A check of the structure of the student's solution then confirms the lack of a base case, which is seen by the fact that there is no conditional statement, in fact no cases that do not make a recursive call.

Diagnosis of a second type of error is illustrated in Example 2.

```
(defproc eldest (x)
        (if (null (children x))
            x
            (eldest (first (children x)))))

Ask away:(eldest Linda)
(Your answer is --)Linda
OUTPUT:(Not far enough up)
DIAGNOSTICS: Brent Brent Marvin Linda
(All names but incorrect - no idea)
STRUCTURE:(The test of the base case does not look ahead)
```

Example 2

In this solution the student has proceeded down the tree rather than up (i.e., **children** has been used rather than **parent**). This results in the *youngest decendent* variant of the problem, in fact. Here, the Eldest Ancestor Task Expert notes that the result is incorrect, so it once again invokes the Eldest Ancestor Diagnostician. The selected cases *Rick*, *Gord*, *Jennifer*, and *Linda*, are run once again and generate answers *Brent*, *Brent*, *Marvin*, and *Linda*, respectively. An observer then notes that that these are all valid names but can find no other rhyme or reason to them (in fact, a more sophisticated observer could have determined that they each appeared lower down in the tree than their corresponding argument). Momentarily baffled, the Diagnostician conducts an examination of the structure of the program (using entities at the program behaviour level) and determines that the "test of the base case does not look ahead" (i.e., does not call **parent**). Hence, structural analysis has succeeded where (at least in this simple example) I/O analysis has failed.

The final example for EA, Example 3, shows the analysis of a correct solution.

```
(defproc eldest (x)
  (if (noparent x)
      x
      (eldest (parent x))))

Ask away:(eldest Rick)
(Your answer is --)Rick
OUTPUT:Rick(-- Answer is correct)
DIAGNOSTICS: Rick Rick Rick Rick
(All test cases are true)
STRUCTURE:(Structure analysis concurs with diagnostics that program
                                                    is correct)
```

Example 3

The analysis of the output of Example 3 indicates that the correct answer is obtained for the supplied input. In order to confirm the correctness of the program, the EA Task Expert calls in the EA Diagnostician. The program runs correctly on the four selected test cases, and the analysis of the program's structure can find no problems either. The Diagnostician thus concludes that this must be a correct program.

In the three examples considered in this subsection the system is able to deal with non-syntactic aspects of attempted solutions to the Eldest Ancestor task. In the next subsection a more complex task is considered.

5.2 Diagnosis in the SST Task

As indicated, the second problem was chosen so that, unlike the single recursive strategy allowed for in the EA task, there could be several possible strategies that a student might employ in his/her solution. Possible strategies for the *Sorted Subtree* (SST) problem, include the following:

1. Traverse the subtree producing a list of the nodes visited. Sort the resulting list. In this strategy the traverse component and the sort component are clearly separate. Any form of traversal (depth-first, breadth-first) is possible, as is any form of sort. In the ensuing discussion, this is referred to as the *traverse-then-sort* strategy.
2. Traverse the subtree. As each node is visited, insert it into its appropriate place in the sorted list. In other words, the sorted list is built as the traversal proceeds, using an insertion sort. This is denoted as the *insert-as-traverse* strategy.
3. In the spirit of the selection sort, traverse the subtree once for *each* node in the subtree. Each traversal is responsible for selecting the next element for the sorted list. This could be termed the *multiple-traversal* strategy.

That these various strategies have advantages and disadvantages is not important at this point. What is important is that they represent three possible strategies that a student might employ in solving this problem. Before any help can be dispensed by the SCENT advisor it must first deduce the strategy employed. Once this is accomplished, specific advice can be given on the implementation of the chosen strategy.

Strategy is determined by a thorough examination of the program -- how it is structured and how it executes. This involves information concerning both static and dynamic aspects of the program that is compiled by the *program behaviour* level of the advisor. A *function chart* gives a static representation of the call-return hierarchy employed in the program code, and is useful in conveying the structure of the solution to observers and other higher level entities. Figure 3 shows a function chart obtained for a solution to the SST task, in both textual and graphical formats. Information about the program's dynamic behaviour is assembled in the form of traces. These come in a variety of forms, the most important of which are the *who-calls-me (WCM)* trace and the *whom-I-call (WIC)* trace. Each of these is an abstraction of the basic sequential execution trace, designed to organize the information in a more useful way. The WCM trace is useful for summarizing how other functions use a given function; in particular, how inputs to the function are transformed by the function. Essentially a WCM trace allows data flow analysis to be carried out. The WIC trace is useful for finding patterns of function calls employed by the function in achieving its objective. In other words, the WIC trace contains dynamic control flow information not available in the function chart. Detailed examples of these traces are given in [McCalla, Bunt, and Harms 1985].

```
-----------------------------------------------
FUNCTION CHART:

sst
  USER-DEFINED --> (sort dftraverse)
  KINTREE PRIMITIVE --> nil
  SYSTEM PRIMITIVE --> nil
sort
  USER-DEFINED --> (sort insert-one)
  KINTREE PRIMITIVE --> nil
  SYSTEM PRIMITIVE --> (if null butfirst first)
dftraverse
  USER-DEFINED --> (dftraverse)
  KINTREE PRIMITIVE --> (children)
  SYSTEM PRIMITIVE --> (null if atom mcons append first butfirst)
insert-one
  USER-DEFINED --> (insert-one)
  KINTREE PRIMITIVE --> nil
  SYSTEM PRIMITIVE --> (null list if alphalessp mcons first butfirst)
-----------------------------------------------
```

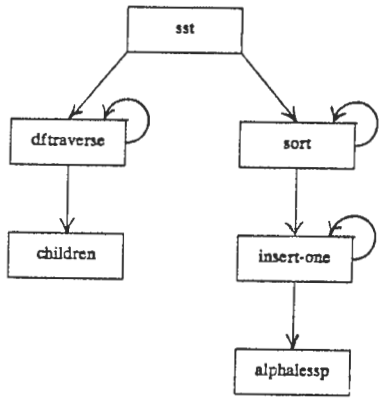


Fig. 3: The Function Chart

Deciding what strategy a student is attempting to implement is accomplished through *strategy judges*. A strategy judge tries to recognize whether or not the solution attempted implements a particular strategy by identifying the functional parts of the task and the way in which they are put together. Strategy judges exist, therefore, for each strategy a student could employ in solving the task. The functional parts of the solution correspond to subtasks that may have their own task expert. For example, in the SST task, the Traverse-then-Sort Strategy Judge may recognize the solution strategy, and identify the Sort and Traverse components. The Sort Task Expert can then analyze this subtask separately to determine the sort strategy used and any errors that might be made in its implementation, in a manner like that used for the EA task.

A student's possible solution to the SST task is given as Example 4.

```
(defproc sst (root)
  (sort (dftraverse root) nil))

(defproc dftraverse (x)
  (if (null x)
      nil
      (if (atom x)
        (cons x (dftraverse (children x)))
        (cons (first x)
              (append (dftraverse (children (first x)))
                (dftraverse (butfirst x))))
      )
  ))

(defproc sort (ulist slist)
  (if (equal (findsmallest ulist slist ZZZ) ZZZ)
      slist
      (sort ulist (append1 slist (findsmallest ulist slist ZZZ)))
  ))

(defproc findsmallest (ulist slist min)
  (if (null ulist)
      min
      (if (and (alphalessp (first ulist) min)
               (null (member (first ulist) slist)))
        (findsmallest (butfirst ulist) slist (first ulist))
        (findsmallest (butfirst ulist) slist min)
      )))
```

Example 4

This particular program consists of four user-defined functions: **dftraverse** performs a depth-first traversal of a tree; **sort** performs a selection sort; **findsmallest** is part of the sort subtask and returns the smallest element of a list; and **sst** drives the subtasks of sort and traverse. The strategy behind this solution is to traverse the entire tree and then sort that list (i.e., the traverse-then-sort strategy). This program contains no errors. Note that the detection of the parts of the solution cannot rely on suggestive function names. In an actual student program **sort** could well have been called **order**, **fn1**, **albert**, etc.

Knowing that the student has chosen to work on the SST task, the system calls on the SST Task Expert, which in turn requests the various traces and a function chart. The input to the SST task is the name of the root node of the subtree that is to be sorted. Running this solution on the input "Darwyn" (see Fig. 2) yields the answer

(Andrea Darwyn Jennifer John Marvin)

which is the correct result. If an incorrect result is obtained, the output may indicate which part of the solution contains the error. Even if the expert decides that the solution is correct without help from strategy judges and diagnosticians, it would probably try to discover which strategy the student used so it can indicate how efficient that chosen strategy is, or how effective the student's solution using that strategy is. This information might be useful to the student knowledge component.

This solution follows the traverse-then-sort strategy, as will be discovered by the Traverse-then-Sort Strategy Judge once it is invoked. Strategy judges for other possible strategies may also be invoked, but their deliberations will ultimately fail, so they will be ignored for this discussion. The recognition of the strategy proceeds in the following way. The function chart (see Fig. 3) indicates that the Sort and Dftraverse parts of the solution are independent of each other. These components are detected by the appearance of **alphalessp** and **children**, respectively, in the function chart. The appearance of **alphalessp** indicates the comparison part of the sort; the rest of the sort in the traverse-then-sort strategy is assumed to be the same leg of the function chart as that containing the function **sort** (not including the driver - **sst**).

Once the sort and traverse subtasks are identified, the Sort Task Expert and the Traverse Task Expert can be invoked on the applicable parts of the code. The Traverse-then-Sort strategy for SST is particularly well-suited to employing experts for the subtasks because these subtasks are completely independent of each other. In this situation, the Selection Sort Strategy Judge identifies the sort as a selection sort. It does this by combining the observations of a number of observers. In particular it finds a selection routine in the solution by finding a function that makes a comparison (i.e., calls **alphalessp**) and produces a single output element (atom type output) which is smaller than or equal to the smallest element of one of the function's list parameters. The driver to the selection function builds up the sorted list by inserting the next smallest element found by the selection sort. This strategy judge may also look for a sentinel value in the comparison test. A sentinel, ZZZ, appears in this solution but is not a necessary condition for a selection sort since some selection sort solutions do not use one.

The strategy judges must be flexible enough to handle variations in the solutions. For instance, there are many different ways of implementing a selection sort because there are many different ways of ensuring that the items inserted in the sorted list are not selected again (i.e., many different ways of marking elements already seen).

Neither the Selection Sort Diagnostician nor the SST Traverse-then-Sort Diagnostician can find an error in the solution. The SST Task Expert is able to report that the solution to SST is correct and that the solution used the traverse-then-sort SST strategy, a selection sort, and a depth-first traversal. Additional comments may be warranted; for example the traverse-then-sort strategy is less efficient than the insert-as-traverse strategy but it is also more modular. The task expert may also report this to the blackboard. Different versions of strategies may be more efficient or less efficient as well.

The approach described in Example 4 provides a framework for the detection of errors in strategy. For example, a student may have an error in a subtask part of his/her solution. As Example 5, consider the following erroneous code for a selection sort. This portion of the code would be called in by the SST solution above, replacing the former sort subtask consisting of **sort** and **findsmallest** functions.

```
(defproc sort (list)
  (if (null list)
        nil
        (mcons (smallest list ZZZ)
          (sort list))
  ))

(defproc smallest (list min)
  (if (null list)
        min
        (if (alphalessp (first list) min)
          (smallest (butfirst list) (first list))
          (smallest (butfirst list) min)
        )))
```

Example 5

This particular implementation of selection sort does not ensure that the elements it has already selected are not selected again. Thus, the smallest item of the input list is chosen over and over again. Another characteristic of this ''marking'' error is an infinite loop which is detected by too many calls to the selection function. In fact, running the program on the same input as before, ''Darwyn'', yields the result:

(Infinite loop detected)

Once again, the output indicates there is an error, but it gives little indication as to where the error may be or why it was made.

Diagnosis proceeds as before. The SST Task Expert invokes the strategy judges. The Traverse-then-Sort Strategy Judge recognizes the strategy as described. The Sort and Traverse parts of the solution are identified and task experts for these subtasks are invoked. The Traversal Task Expert executes the traversal part of the program on its own data and finds that the traversal is working. The Sort Task Expert executes the sort part of the solution and obtains the same infinite loop error message as the SST Task Expert received, suggesting that the error is in the sort part of the solution. Attention then turns to this component.

Diagnosis of this subtask is summarized by the following output.

```
Ask away:(sl1-df Darwyn)
SST answer is -- (Error in kineval - infinite loop detected)
SST STRATEGY 1
        SST1 Sort: sort    SST1 Traverse: dftraverse
-------------------------------------------------------------------

SORT parameters are -- ((Darwyn Jennifer Marvin John Andrea))
SORT input is -- (Darwyn Jennifer Marvin John Andrea)
SORT answer is -- (Error in kineval - infinite loop detected)
SORT STRATEGY (select)
        Sort Select Drivers nil      Sort Selecter (smallest)
        (sort SELECT DRIVER does not break down by smallest or build up
SORT SELECT DIAGNOSTICIAN -                                  by largest)
        (Wrong select answer - smallest)
        (There appears to be problems with marking already sorted items)
-------------------------------------------------------------------

SST DIAGNOSTICIAN -
        Wrong SST answer
        Error is in sort
                (Wrong select answer - smallest)
                (There appears to be problems with marking already sorted
                                                              items)
```

Example 5a

The Selection Sort Strategy Judge detects the selection strategy by finding the selection function. In this case, the driver does not build the sorted list up at all since only one item is ever selected. The selection function is found, however, because the smallest element is always chosen. There is also a sentinel value, ''ZZZ'', found.

The Selection Sort Diagnostician deduces that there is a ''marking'' error. The clues are the infinite loop error message and the fact that the selection function always returns the same answer, namely the smallest element of the original input. The Equal Output Observer checks if all of a function's output is equal to some value and returns this value. The diagnostician can then compare this value to the input to see if it is the smallest element. The Sort Task Expert reports this ''marking'' error to the blackboard.

Next, the SST Traverse-then-Sort Diagnostician is invoked, and, using the information from both the Sort and Traversal task experts, it indicates that there is a marking error in the sort subtask. The SST Task Expert reports this, while the diagnostician could look for other errors (e.g., the interface between the sort and traverse subtasks).

6. Informing the Student

Clearly an important part of the system is the way in which it reports its findings to the student. This calls upon the *Student Model* component, a part of the system which has seen a lot of thinking but, unfortunately, little in the way of implementation at the present time.

Ultimately, all diagnoses must be further interpreted in light of knowledge about the particular student who wrote the program. This will be necessary in order to provide relevant feedback to the student. A problem with misplacing the base case in a recursive solution to the eldest ancestor task, for example, may be merely an accident, may indicate a fundamental misunderstanding of recursion, or may indicate difficulties with the **if** statement. Determining which of the possible explanations is most appropriate would require knowledge about the student's previous behaviour in order to distinguish the kinds of concepts which the student seems to know from those he/she hasn't practised or those in which he/she has demonstrated weaknesses. The collection, maintenance, synthesis, and communication of this kind of knowledge is the responsibility of the student model. Much more work needs to be done in this area and is underway.

7. Conclusion

This paper has described the role of the programming advisor component of the SCENT Student Computing Environment in analyzing student programs and dispensing advice to students concerning problems with their solutions. Of particular interest is the handling of non-syntactic errors. Central to the approach is the notion of task experts which rely on knowledge of the domain in which the student is working and the particular task he/she has selected to define the necessary context within which to interpret attempted solutions. The diagnosis of errors relies on information obtained from analyzing both the static structural properties of

the solution and its dynamic execution behaviour. The use of these was illustrated by means of discussion of detailed examples.

In its current state of development SCENT has already proven to be interesting. One of its major contributions has been to provide a multi-layered architecture with roles for low-level program knowledge, domain and task-specific knowledge, and student knowledge. In particular the kinds of low-level knowledge used go beyond most other approaches and include trace information, cross-reference charts, etc., to be used in addition to the student's code. The concept of observers which interpret this information to the higher-level processes is unique and is a powerful way of organizing the multiplicity of low-level information produced by the task-independent components of the SCENT advisor. The domain and task-specific knowledge is organized into natural divisions which clearly separate task knowledge from knowledge of strategies and knowledge of various bugs. Student knowledge, although not elaborated on in the current system, can be easily incorporated into future SCENT expansions without disruption to the rest of the system. The blackboard communication mechanism promises to be an interesting vehicle for exploring knowledge-based distributed control and information passing.

8. Acknowledgements

Discussions with Marlene Jones, of the University of Waterloo, and Daniel Zlatin, of Bell Northern Research, aided in the formation of some key ideas in this work. Various insights have been provided by Bryce Bairie, Judy Escott, Xueming Huang, Ken McDonald, Teng Ng, Paul Pospisil, and Berni Schiefer, most of whom continue to work on the SCENT project. The research was supported, in part, by the Natural Sciences and Engineering Research Council of Canada, through operating grants numbered A3592 and A3707.

9. References

[1] Abelson, H., Sussman, G.J., and Sussman, J., *Structure and Interpretation of Computer Programs*, The MIT Press, McGraw-Hill Book Company, 1985.

[2] Adam, A. and Laurent, J.P., "LAURA, A System to Debug Student Programs", *Artificial Intelligence Journal*, Vol. 15, 1980, 75-122.

[3] Anderson, J.R. and Reiser, B.J., "The LISP Tutor", *BYTE*, Vol. 10, No. 4, April 1985, 159-175.

[4] Erman, L.D., Hayes-Roth, F., Lesser, V.R., and Reddy, D.R., "The HEARSAY II Speech-Understanding System : Integrating Knowledge to Resolve Uncertainty", *Computing Surveys*, Vol. 12, No. 2, 1980, 213-253.

[5] McCalla, G.I. and Murtagh, K.M., "GENIUS: An Experiment in Automated Program Advising", Research Report No. 85-20, Department of Computational Science, University of Saskatchewan, December 1985.

[6] McCalla, G.I., Bunt, R.B., and Harms, J.J., "The Design of the SCENT Automated Advisor", Research Report No. 85-22, Department of Computational Science, University of Saskatchewan, December 1985.

[7] Rich, C. and Shrobe, H.E., "Initial Report on a LISP Programmer's Apprentice", *IEEE Transactions on Software Engineering*, Vol. 4, No. 6, November 1978, 456-467.

[8] Soloway, E. and Johnson, W.L., "PROUST: Knowledge-Based Program Understanding", *IEEE Transactions on Software Engineering*, Vol. 11, No. 3, 1985, 267-275.

[9] Soloway, E., Woolf, B., Rubin, E., and Barth, P., "MENO-II: An Intelligent Tutoring System for Novice Programmers", *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, British Columbia, 1981, 975-977.

[10] Wertz, H., "Stereotyped Program Debugging: An Aid for Novice Programmers", *International Journal of Man-Machine Studies*, Vol. 16, 1982, 379-392.

Using Relative Velocity Information To Constrain The Motion Correspondence Problem: Psychophysical Data And A Computational Model

Michael Dawson and Zenon Pylyshyn

Centre for Cognitive Science
The University of Western Ontario
London, Ontario, Canada

ABSTRACT

This research examined how the human visual system solves the motion correspondence problem. The results of three psychophysical experiments demonstrated that the visual system is sensitive to interdependencies between the motion of different elements when motion correspondence is computed, which is contrary to the independence hypothesis in Ullman's [27] minimal mapping theory. A computational model that solves the motion correspondence model by minimizing the relative velocities of elements was developed. For a wide variety of apparent motion displays, this model generated motion correspondence matches that were consistent with those generated by the human visual system.

Introduction

The visual system is capable of generating the perception of movement from two static images. In producing this illusory or apparent motion, the visual system must maintain the identities of image elements over time. It must be capable of identifying an element in a location in the first image (Frame I) and another element in a different location in the second image (Frame II) as constituting different glimpses of the same moving element. A *motion correspondence match* between a Frame I and a Frame II element is such an identification.

One might expect that motion correspondence matches are easily determined on the basis of figural information. For instance, figural processing could be used to draw motion correspondence matches between look-alike elements in the two frames of view (e.g., [21]). However, the human visual system does not appear to operate in this fashion. It will often make motion correspondence matches between objects of different shapes or colours (e.g., [15], [16], [17], [20]). Indeed, the fact that apparent motion can be perceived in displays consisting of elements that cannot be figurally distinguished (e.g., a number of small dots) suggests that the human visual system computes motion correspondence matches solely on the basis of element position information.

The visual system computes unique apparent motion percepts for displays that have more than one logically possible interpretation. For example, consider the display presented in Figure 1(a). Human subjects generally see the pattern of motion depicted on the left. The pattern of motion on the right is logically possible, but is not typically perceived. As more elements are added to the display, the number of logically possible interpretations that are *not* computed increases dramatically, as is shown in Figure 1(b). If there are N elements in the two frames of an apparent motion display, then there are $N!$ different sets of motion correspondence matches that are logically possible. Ullman [27] has called the situation in which a single set of motion correspondence matches must be selected from a number of logically possible sets the *motion correspondence problem*. The discovery of the rules that the human visual system uses to select sets of motion correspondence matches is a major concern of apparent motion research.

The purpose of this paper is to describe a computational model of how the visual system might solve the motion correspondence problem. This model is based on the assumption that a simple constraint is applied to the problem that prevents the selection of any set of motion correspondence matches that is not the correct set. This paper proceeds as follows. First, some previous work on the motion correspondence problem is briefly reviewed. Then the logic behind the *relative velocity constraint*, which can be used to constrain the motion correspondence problem, is developed. Third, a description of a parallel computation that implements this constraint in a simple computational network is provided. Finally, the results of simulation tests of this model are described and compared with empirical findings. These results demonstrate that the relative velocity constraint is a viable means of solving the motion correspondence problem.

Ullman's Approach to the Problem

In visual perception, there often arise situations in which more than one percept could logically be computed from the same pattern of proximal stimulation. The motion correspondence problem is but one example of this type of situation. Researchers following the natural computation approach to vision have had considerable success in addressing problems of this type by discovering particular types of restraints that can account for why we perceive some patterns, and why we do not perceive other logically possible patterns. The restraints discovered by applying the natural computation approach tend to all be of one particular type: they can be characterized in terms of common, objective properties of the world. For example, Ullman [27] has used the property that most objects in the world are rigid to constrain the computation of the three-dimensional structure of objects from different two-dimensional views of the positions of object elements. A property of the world that can be used to constrain the possibilities in some underdetermined perceptual task is called a *natural constraint*. Natural constraints are an important type of restraint, because they are based on general physical properties, and are therefore universally true of the world. Marr ([19], p. 331) argues that "the discovery of valid, sufficiently universal constraints leads to conclusions about vision that have the same permanence as conclusions in other branches of science".

In applying the natural computation approach to the motion correspondence problem, an attempt is made to discover one or more natural constraints that can explain why some sets of motion

correspondence matches are preferred over other logically possible sets of matches. Ullman ([27], chpt. 3) provides an example of this approach to the problem with his minimal mapping theory of motion correspondence. He begins by assuming that every motion of an element in the world has an associated probability of non-occurrence. In general, when a moving three-dimensional image is projected onto the two dimensional retina, slow movements are more likely to be seen than fast movements. The negative logarithm of the probability of non-occurrence is called the *cost* of the element's motion. The cost of the movement of an element is used to constrain the motion correspondence problem. The visual system selects the one set of motion correspondence matches from the *N!* possible sets of matches that has the smallest total cost. This is equivalent to selecting the set of correspondence matches with the highest overall probability of occurrence.

Two further assumptions are required to allow a system to use the cost constraint to successfully solve the motion correspondence problem. It is assumed that the cost of one possible element motion in a display is independent of the cost of other other possible element motions in the display. This in turn means that the total cost of a particular set of correspondence matches is the sum of the individual costs associated with each match in the set. An additional assumption must be made to force the system to make motion correspondence matches, because the least-cost correspondence solution is the one in which no motion correspondence matches have been made at all (i.e., all Frame I elements are interpreted as disappearing). Ullman [27] avoids this difficulty by assuming that the visual system operates under what he calls the *cover principle*. The cover principle is the assumption that each element in Frame I must be matched with at least one element in Frame II, and that each element in Frame II is matched with at least one element in Frame I. This is the same as assuming that Frame I elements will not suddenly disappear, and that Frame II elements will not suddenly appear. Use of the cover principle prevents the computed least-cost correspondence solution from containing no correspondence matches at all.

Ullman [27] has shown that a simple parallel network that takes as input the positions of elements in Frames I and II, and that applies the minimal cost constraint under the additional assumptions of independence and the cover principle, will generate unique solutions to the correspondence problem that usually are in agreement with human perceptions of apparent motion displays. This network has three different types of operators that take input from a small number of neighbouring operators. One type of operator is used to find motion correspondence matches of minimum cost. The other two types of operators ensure that the cover principle holds for both Frame I and Frame II display elements. The successful performance of his model indicates that the restraints that he has applied to the motion correspondence problem are particularly apt. However, the model has faced some criticism. For example, Marr ([19], p. 201) argues against the use of probabilistic constrants, such as cost. Rogers [24] suggests that some of the constraints in the model (for instance, the cover principle) seem more related to experimental regularities than to properties of the physical world -- contrary to the "natural constraints" principle. A major goal of the present research was to see if an alternative model could be formulated that did not suffer from these problems.

Testing the Independence Hypothesis

Ullman (e.g., [26], [27]) has used the *motion competition paradigm* to investigate motion correspondence processing empirically. In this paradigm a central Frame I element is displayed in succession with two lateral Frame II elements, as is depicted in Figure 2(a). The distance between the Frame I and each of the Frame II elements is usually varied in an experiment. Subjects are asked to indicate the direction of movement (left or right) of the Frame I element; this measures which of the two possible motion correspondence matches is computed by the visual system. Usually, the central Frame I element is seen to move in the direction of the nearest Frame II element. The question of interest, with respect to the independence hypothesis, is whether this is the case when other moving elements are added to the standard motion competition display.

Ullman ([27], p. 84-85) found that subjects' perceptions of the direction of motion of the central Frame I element were *not* affected when an additional pattern of motion was added to the critical display, as is depicted in Figure 2(b). Ullman used this result to defend the independence hypothesis. However, this hypothesis requires additional investigation. This is because there is a growing literature that suggests that motion perception is sensitive to the movement of elements relative to one another (e.g., [9], [12], [13], [18], [22], [23]). Whether this sensitivity emerges *after* motion correspondence is computed, and is used to interpolate the appearance of objects as they move, or whether this sensitivity can be found to affect the motion correspondence process, is an empirical issue. Three experiments were run to examine this issue by using a variation of Ullman's display, such as the one presented in Figure 2(c).

Experiment I

The purpose of Experiment I was to test the validity of the independence hypothesis by using the motion competition paradigm. Motion competition displays that consisted of a black central vertical line in Frame I put into competition with two black lateral vertical lines in Frame II were used. The ratio of the distances between the two lateral elements and the central element was independently varied. Four white horizontal lines, two above and two below the competition display, were used to provide an apparent motion context. Figural grouping of the competition and context elements was prevented by making them opposite in contrast and orientation (c.f., [3], [30]), and by minimizing element collinearity. Five different context conditions were used in the experiment. In the two experimental conditions, all four context elements moved in the same direction (left or right), moving as far as the appropriate Frame II element in the competition display. In two context control conditions, the upper and lower context elements moved in opposite directions. In a third control condition, no context elements were used at all. Instances of all conditions were randomly presented to two observers whose task was to indicate whether the Frame I competion display element moved to the left or to the right. Each trial consisted of a single Frame I to Frame II alternation. In order to control for the effect of eye movements, subjects were instructed to fixate on the centre of the display (which was marked by two fixation crosses), and the stimulus onset asynchrony was set to 100 ms.

A psychophysical function that related the probability of seeing the central competition display element move to the left as a function of interelement distance was plotted for each subject and each condition. These functions are illustrated in Figure 3. Under the independence hypothesis, it would be predicted that the five functions for each subject should be identical. This was *not* the case. For both subjects, the three control functions were nearly identical, and indicated that the threshold for the motion detection task (i.e., the interelement distance at which the Frame I competition element is seen to move to the left or to the right with equal probability)

occurred when the two Frame II elements were approximately equally displaced from the Frame I element. The two experimental condition functions were quite different. When the context elements moved to the right, the resulting threshold function was shifted to the right from the control functions. This indicated that for this condition, it was more likely to see the competition element move in the same direction as the context. Similarly, when the context elements moved to the left, the resulting threshold function was shifted to the left from the control functions, which indicated an increased tendency to see the competition element move to the left.

Experiment II

One potential problem in the design of Experiment I was that it was possible to perceive a moving, rigid combination of context and competition display elements. As a result, it is possible that the observed effects of motion context were due to figural grouping processes, in which a rigid configuration of the context and the competition display was seen to move as a figural whole, and were not due to the interdependence of the *motion* of the elements. Although this does represent a failure of the independence hypothesis, the question still remains whether the context effect is found without the coherent movement of a pattern that includes the competition elements. One way to rule this possibility out is to show that the effect occurs when the context and the central competition elements move in the same directions but at different speeds. Experiment II was designed to determine if this was the case. It was also an attempt to see if the motion context effect could be strengthened by adding more elements to the context.

In this experiment a true method of constants design was used to create stimuli very similar to those used in Experiment I. On each trial, one of the two Frame II competition display elements was randomly chosen to be a standard distance of 270′ from the central Frame I element. The other Frame II competition display element was displaced away from the Frame I element by one of ten comparison distances. Five of the comparison distances were shorter than the standard in decrements of 11.3′ of visual angle. The other five comparison distances were longer than the standard in increments of the same amount.

In two thirds of the trials, the competition display described above was presented together with one of four moving contexts, consisting of from one to four pairs of elements. On each trial the context would be shifted in position 270′ (the standard distance) to the right or to the left. In half of these trials, the context elements moved in the direction towards the Frame II element that was located the standard distance from the center of the competition display (i.e., the standard direction). These trials provided a replication of Experiment I, for if the central competition display element was seen to move in the direction towards the context, it would have the same velocity as the context. In the other half of these trials, the context moved in the same direction as the Frame II element that was located the comparison distance from the center of the competition display (i.e., the comparison direction). If the central competition display element was seen to move in the same direction as the context in these trials, then it would *not* be moving the same distance as the context, because the context always moved the standard distance. In these trials, the difference in distances travelled ranged from the context travelling 56′ of visual angle less than the competition element to the context travelling 56′ more than the competition element. Therefore, in these trials the context and the competition display could not be grouped into a rigid figural whole. In the other third of the trials, subjects saw a control display in which no context elements were present. Three subjects participated in the study. Again, their task was to detect the direction of motion of the Frame I competition display element. Subjects fixated on the centre of the display, and the SOA was 150 ms.

The data from this experiment was analyzed by determining the threshold interelement distance (relative to the standard) at which the central motion competition element was seen to move left or right with equal probability [4]. For each subject and for each context condition three thresholds were computed: one for the no context control, one for trials in which the context moved in the standard direction, and one for trials in which the context moved in the comparison direction. The threshold for the no context control trials serves as a baseline measure for each subject in each condition. If perceptions of the competition display depend upon the motion of the context, then the following predictions can be made: In trials in which the context moves in the standard direction, preference for seeing motion in the comparison direction should decline. As a result, the thresholds computed for these trials should be smaller than those computed for the baseline trials. Similarily, in trials in which the context moves in the comparison direction, preference for seeing motion in the comparison direction should increase. As a result, the thresholds computed for these trials should be larger than those computed for the baseline trials.

The results of Experiment II strongly supported the above predictions. Across all of the subjects, the average baseline threshold was 278′ of visual angle, while this threshold was 197.44′ when the context moved in the standard direction, and was 491.17′ when the context moved in the comparison direction. This replicates the findings of Experiment I. Also, as this effect is observed in trials in which the context and the competition elements cannot be grouped into a rigid whole, these results indicate that the observed interdependence involves the movement of individual elements, and not figural grouping processes.

The results do not demonstrate a very strong relationship between the size of the motion context element effect and the number of elements in the context. The correlation between the threshold difference from baseline and the number of context elements is -0.11 for the standard direction context and is -0.07 for the comparison direction context. One reason for this small relationship may be that subjects were exposed to only one level of this manipulation per block of trials. The relative strengths of these different types of contexts might best be examined in a completely randomized design. Also, as more elements were added to the contexts, these elements were placed further away from the competition display. Hence there is a confound between the number of elements in a context and their average proximity to the critical display. A third experiment was run to determine whether context proximity affected performance on the direction detection task.

Experiment III

Experiment III was designed to test whether the proximity of context elements to the competition display affected the strength of their effect on the direction detection task. The display parameters of this study were nearly identical to those used in Experiment II. However, a different context manipulation was used. In Experiment III, all motion contexts consisted of two pairs of elements (one pair above and one pair below the central display) displaced from the competition display by one of three vertical separations (81.82′, 196.4′, or 310.9′ of visual angle). A no context control condition was used as well. All conditions were completely randomized, and three subjects participated. Thresholds for the direction detection task were calculated for each subject and each condition.

Two principle findings were obtained from this study. First, the motion context effect on the direction detection task was replicated. Averaging over subjects and proximity conditions, the control (no context) threshold was 277.4′, while the threshold was 221.0′ when the context moved in the standard direction and 364.0′ when the context moved in the comparison direction. This pattern indicated a tendency to see the Frame I competition display element move in the same direction as the context. The second finding was of a general proximity effect: the closer the context was to the competition display, the stronger was its effect on the direction detection threshold. When the context moved in the standard direction, the threshold was 191.95′ for the near proximity, was 229.29′ for the middle proximity, and was 241.84′ for the far proximity. When the context moved in the comparison direction, the threshold was 427.74′ for the near, was 341.71′ for the middle proximity, and was 322.55′ for the far proximity. In both cases, displacing the context further away from the competition display produced direction detection thresholds that were closer in magnitude to the control threshold.

Discussion

The results of the three studies described above clearly show that the direction of apparent motion of the central element in the motion competition display is affected by the presence of a moving context. If the context moves to the right, the visual system shows a strong preference to see the central element move to the right. If the context moves to the left, the visual system shows a strong preference to see the central element move to the left. These results are in direct conflict with the independence hypothesis of minimal mapping theory, and indicate that the motion of elements is interdependent as far as motion correspondence is concerned. This raises the question of whether a different motion correspondence model, one that systematically takes interdependence into account, can be developed. In the next section, the idea behind such a model is described. Then a computer simulation of a motion correspondence model that assumes interdependence is detailed.

The Relative Velocity Constraint

The results of the three experiments indicate that element interdependencies are involved when motion correspondence is computed, but do not indicate the precise nature of this interdependency. In accordance with the principle of natural constraints, this should be found by considering properties of the physical structure of the world. For example, motion correspondence appears to be computed over tokens of small parts of objects, such as edge parts, terminators, and intersections ([27], chpt. 2). This suggests that the relationships between the movements of these elements with respect to their arrangement in space should be considered. In particular, it is important to note that the closer two elements are in an image, the more likely are they to be parts of the same object. This is because object boundaries account for a very small proportion of the visual field (e.g., [19], pp. 44-51); as a result it is very unlikely that an object boundary will lie between two elements that are close together in the image. From this it follows that if elements are near one another on a rigid or near-rigid object, then the two elements should have very similar instantaneous movement vectors in three dimensions. Such similarity of movement in three-dimensions projects as similar movement vectors in two dimensions, though the statistical distributions are transformed [5]. As a first approximation, we assume that this constraint favours similar vector velocities in two dimensions when image elements are close together. We therefore constrain the motion correspondence problem in the following way: the motion perception system should select a set of motion correspondence matches such that elements near one another in the image have similar velocities. In other words, the relative velocity (i.e., the difference between two velocities) of local elements in the image is minimized. This is called the *relative velocity constraint*.

It should be noted that similar constraints have been applied to other problems in motion perception. In particular, Hildreth [10] and Horn and Schunk [11] compute retinal velocity fields by assuming that differences between velocities vary smoothly over the entire velocity field. This means that velocity vectors are assumed to be most similar to their neighbours, and less similar to velocity vectors located further away in the velocity field. Our use of the relative velocity constraint differs from this previous work in two respects. First, while the smoothness constraint was designed to be applied to image points that make up continuous contours, the relative velocity constraint is designed to be applied to discrete components of a display. Second, the smoothness constraint was used to solve a problem in which the input is some partial information about the velocity field, which can in principle lead to the computation of an infinite number of complete velocity fields. Using the relative velocity constraint to solve the motion correspondence problem involves very different circumstances. Each display element is initially assigend a number of complete velocity vectors, and this initial state can lead to the computation of only a finite number of motion correspondence solutions. Nevertheless, it is interesting to note that similar constraints may be used to solve different motion perception problems. There is some recent evidence suggesting that the velocity field perception mechanism and the apparent motion system may share common neural mechanisms ([1], [2], [7], [8]). Thus it would not be surprising to find that they operated under some similar constraints.

A Relaxation Labeling Program

Zucker (e.g., [28], [29]) argues that many problems in low-level vision are most appropriately solved by local, parallel networks of visual operators. He describes a particular type of algorithm, called a *relaxation labeling* algorithm, that exhibits these characteristics, and that has been successfully applied to a number of low-level vision problems. In a relaxation labeling algorithm, ambiguous assertions about some state of affairs are disambiguated by discarding assertions that are incompatible with potentially valid assertions about neighbouring states of affairs. The goal of a relaxation labeling algorithm is to compute a set of labels that are locally compatible with each other on some relevant measure. The following sections describe a relaxation labeling program that solves the motion correspondence problem by measuring local compatibility in terms of the relative velocity constraint.

Relaxation labeling is performed over a structure that is best represented as a network of linked nodes. The nodes in this network usually represent specific locations in the visual field. Links between nodes define neighbourhoods in the visual field. Neighbourhood relationships are important, for the properties of a node are determined by the properties of the other nodes in its neighbourhood. The properties of a node are represented as labels that are attached to it. For example, in a low-level vision problem in which one is interested in finding edges of objects, one might assert that "a line segment exists at place z" by attaching the label "line segment" to the node that represents "place z in the visual field".

The relaxation labeling algorithm developed to solve the motion correspondence problem uses a particular form of the network representation described above. The nodes in this network represent the positions of Frame I elements in an apparent motion display. A link between nodes is made if the Frame I elements are neighbours in the visual field. In the current version of the program, two Frame I

elements are considered to be neighbours in the Cartesian coordinate system used if they are no further apart than 5 "units" of distance. (The results of Experiment III suggest that 5 distance "units" in the model would correspond to approximately a degree of visual angle for human observers). The labels assigned to the nodes in this network represent particular motion correspondence matches that can be made between that particular Frame I location and another element in Frame II. In essence, the labels are the "names" of the correspondence matches that could possibly originate from that Frame I location.

In addition to the labels described above, each node is assigned one other label, called a *null label*. The purpose of the null label is to free the network from operating under the constraints of the cover principle. If at the end of computations the null label is assigned to a node in the network, then this indicates that the element corresponding to this position in Frame I has disappeared from view in Frame II. A null label corresponds to an "imaginary" motion vector that is computed by adding together the actual motion vectors that could originate from a particular Frame I location, and then taking a vector opposite in direction but equal in magnitude. The logic behind using this type of null label is that if it is selected over any of the possible correspondence matches that could originate from a node, then the best match to apply to a node is opposite to any that are actually possible. Hence, the element associated with this node should be assigned no motion correspondence matches at all. If this type of null label was not used, then a Frame I element would always have to be matched to a Frame II element. Null labels are not required to model uncovered Frame II elements that suddenly appear in the visual field. These elements are not given a motion correspondence match, indicating that they were not present in Frame I of the display. model.

The algorithm proceeds by iteratively discarding labels (i.e., motion correspondence matches) that are incompatible with neighbouring labels. Each label represents a motion vector. The "difference" between two labels is then taken to be the Euclidean distance between the endpoints of the motion vectors that they represent, after they have been centered to a common origin. The algorithm used such relative velocity "differences" as a criterion to discard labels as follows: A cost is computed for each label, where cost is the average relative velocity associated with the label taken with respect to the other labels in the neighbourhood. With each iteration of the network the cost of each label is recomputed, and at each node the label with the highest cost is discarded. If more than one label has the highest cost, then all of these labels are discarded. This procedure continues iteratively until no more labels can be discarded withough leaving nodes unlabelled. This discrete algorithm is very fast: if there are M Frame II elements, then it will converge in no more than M iterations. This is because in the worst possible case, each node in the network will still lose one label with each iteration.

A computer implementation of the model described above was tested. The program takes as input the Cartesian coordinates of the Frame I and Frame II elements in an apparent motion display. It uses this information to compute all neighbourhood links between Frame I element positions, the logically possible motion correspondence labels, and the null labels. In the initial state of the program, all Frame I to Frame II correspondence matches are considered. Labels are iteratively discarded according to the procedure described above until the network converges.

The simplest type of display for the model to deal with is the pure translation of configurations of elements. For this type of display, people perceive all elements as moving at identical velocities. This type of interpretation is also computed by the model (Figure 4(a)). The model will also generate the correct set of motion correspondence matches when elements are *not* translated at identical velocities (Figure 4(b)). When given representations of displays similar to those presented human subjects in the three experiments that were reported, the model emulates human performance (Figure 4(c)). This shows that the null label convention has freed the model from the constraints imposed by the cover principle. Without the null label, Frame I elements that are seen by humans to disappear would be seen by the model to move. The model also generates the group percepts reported by humans for Ternus-like translations (Figure 4(d)), as well as the symmetric patterns of motion correspondence matches (Figure 4(e)) raised as a challenge to apparent motion theories by Kolers [14].

The model produces very interesting solutions for displays in which configurations of elements are rotated about the origin of the image plane. Figure 5 demonstrates this performance for various rotations of three collinear elements and for a square configuration of elements. Note that the trajectory that would result from the computed correspondence matches changes as a function of the amount of rotation: from the perspective of the origin, the resultant trajectory is convex for rotations less than 90°, and is concave for rotations greater than 90°. This finding is consistent with the results of Shepard and Judd [25], who found that the visual system prefers the least-energy path in apparent rotation of complete polygons. Preliminary results from our own experiments that use discrete element diplays like those in Figure 5 indicate that human observers make the same correspondence matches as those computed by the model.

Figure 6 demonstrates that the model is also capable of providing the correct motion correspondence solutions for looming patterns, whether diagonal elements in the square configurations used are neighbours or not. A previous model of ours, which attempted to constrain the motion correspondence problem by only minimizing local differences in the direction of motion, could not solve this type of display [6]. This indicates the appropriateness of the relative velocity constraint, which minimizes differences in the magnitude of element motion as well as direction.

In summary, a computer implementation of the relative velocity constraint in a local, parallel network is capable of solving the motion correspondence problem without requiring the cover principle. The performance of the program for a wide variety of displays is very similar to that of the human visual system. This is particularly true for apparent rotations in the image plane, which lead to least-energy interpretations very similar to those observed in previous research on apparent rotational movement.

Summary and Discussion

Ullman ([27], pp. 104-108) argues that in addition to his empirical demonstration, there are principled reasons to adopt the independence hypothesis. He shows that his minimal mapping model is capable of detection flow patterns of motion, and argues that assuming interdependencies to compute motion correspondence would be redundant. The results presented in the current paper raise questions about the empirical and the theoretical support for the independence hypothesis. The three experiments provide strong evidence that the motion correspondence process computes a motion correspondence match for an element by using information concerning the motion of its neigbours in an attempt to maximize the local similarity of motion. From a theoretical point of view, a motion correspondence model that systematically takes this observed interdependence into account can be developed. While the

assumption of interdependencies may be redundant within the minimal mapping perspective, it is clear that such an assumption can lead to a plausible and empirically tenable alternative motion correspondence model.

References

[1] Anstis, S.A., Giaschi, D., & Cogan, A. (1985). Adaptation to apparent motion. *Vision Researeh, 25,* 1051-1062.

[2] Barbur, J.L. (1981). Subthreshold addition of real and apparent motion. *Vision Research, 21,* 557-564.

[3] Beck, J. (1966). Effect of orientation and of shape similarity on perceptual grouping. *Perception & Psychophysics, 2,* 300-302.

[4] Bock, R., & Jones, L. (1968). *The Measurement and Prediction of Judgement and Choice.* San Francisco: Holden-Day.

[5] Dawson, M. (in preparation). *Using Relative Velocity as a Natural Constraint on the Motion Correspondence Problem.* Doctoral disseration, The University of Western Ontario.

[6] Dawson, M., & Pylyshyn, Z. (1985). The natural computation of motion correspondence. U.W.O. Centre for Cognitive Science Technical Memorandum No. 20.

[7] Green, M. (1983). Inhibition and facilitation of apparent motion by real motion. *Vision Research, 23,* 861-865.

[8] Green, M., & Von Grunau, M. (1983). Real and apparent motion: One mechanism or two? *Proceedings of the SIGGRAPH/SIGART Interdisciplinary Workshop on Motion: Representation and Perception.*

[9] Gogel, W. (1974). The adjacency principle in visual perception. *Quarterly Journal of Experimental Psychology, 26,* 425-437.

[10] Hildreth, E. (1983). *The Measurement of Visual Motion.* Cambridge, Mass.: MIT Press.

[11] Horn, B.K.P., & Schunk, B.B. (1981). Determining optical flow. *Artificial Intelligence, 17,* 185-203.

[12] Johansson, G. (1950). *Configurations In Event Perception.* Uppsala, Sweden: Almqvist and Wiksell.

[13] Kaji, S., & Kawabata, N. (1985). Neural interactions of two moving patterns in the direction and orientation domain in the complex cells of cat's visual cortex. *Vision Research, 25(6),* 749-753.

[14] Kolers, P.A. (1972). *Aspects of Motion Perception.* Toronto: Pergammon Press.

[15] Kolers, P.A., & Green, M. (1984). Color logic of apparent motion. *Perception, 13,* 149-254.

[16] Kolers, P.A., & Pomerantz, J. (1971). Figural change in apparent motion. *Journal of Experimental Psychology, 87,* 99-108.

[17] Kolers, P.A., & von Grunau, M. (1976). Shape and colour in apparent motion. *Vision Research, 16,* 329-335.

[18] Krumhansl, C.L. (1984). Independent processing of visual form and motion. *Perception, 13,* 535-546.

[19] Marr, D. (1982). *Vision.* San Francisco: W.H. Freeman & Co.

[20] Navon, D. (1976). Irrelevance of figural identity for resolving ambiguities in apparent motion. *Journal of Experimental Psychology: Human Perception and Performance, 2,* 130-138.

[21] Price, K.E. (1985). Relaxation matching techniques -- A comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-7,* 617-623.

[22] Proffitt, D., & Cutting, J. (1979). Perceiving the centroid of configurations on a rolling wheel. *Perception and Psychophysics, 25,* 389-398.

[23] Proffitt, D., & Cutting, J. (1980). An invariant for wheel-generated motions and the logic of its determination. *Perception, 9,* 435-449.

[24] Rogers, B. (1980). Review of *The Interpretation of Visual Motion* by S. Ullman. *Perception, 9,* 485-487.

[25] Shepard, R.N., & Judd, S.A. (1976). Perceptual illusion of rotation of three-dimensional objects. *Science, 191,* 952-954.

[26] Ullman, S. (1978). Two-dimensionality of the correspondence process in apparent motion. *Perception, 7,* 683-693.

[27] Ullman, S. (1979). *The Interpretation of Visual Motion.* Cambridge, Mass.: MIT Press.

[28] Zucker, S. (1976). Relaxation labeling and the reduction of local ambiguities. In C. H. Chen (Ed.) *Pattern Recognition and Artificial Intelligence.* New York: Academic Press.

[29] Zucker, S. (1978). Local structure, consistency, and continuous relaxaton. McGill University Computer Vision and Graphics Laboratory Technical Report #78-11R.

[30] Zucker, S., Stevens, K., & Sander, P. (1983). The relation between proximity and brightness similarity in dot patterns. *Perception & Psychophysics, 34,* 513-522.

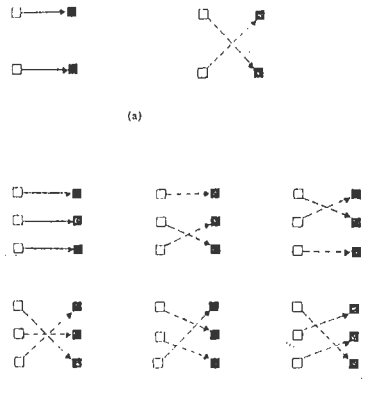


Figure 1. Underdetermination of motion correspondence. Observers make the matches indicated by solid arrows, but matches indicated by dashed arrows are logically possible.

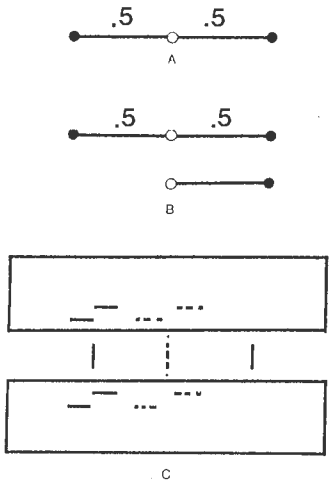


Figure 2. Ullman's (1979) test of the independence assumption. In the motion competition display in (a), the central Frame I element is seen to move to either of the two equidistant Frame II elements with the same probability. This is also true in (b) when an additional moving element is added to the display. A schematic representation of the type of display used in our studies is illustrated in (c). Dotted elements are in Frame I, while solid elements are in Frame II. In the actual displays, the context elements were white bars presented on a black background.

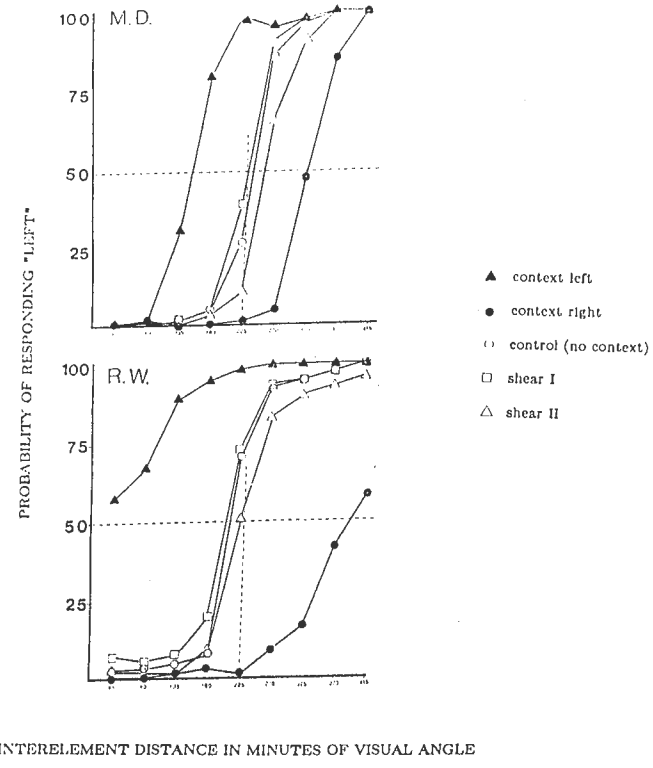


Figure 3. The results of Experiment I for two subjects. The functions depict the probability of seeing the Frame I competition element move to the left as a function of the distance between this element and the Frame II element on the right. Each data point is based on 60 observations.

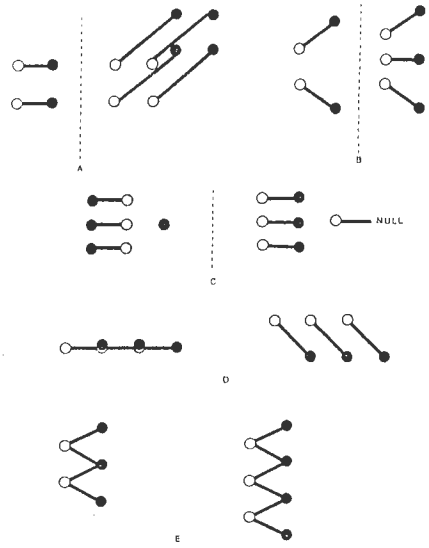


Figure 4. Performance of the model. Open circles are Frame I elements, filled circles are Frame II elements, and solid lines indicate computed motion correspondence matches. (a) Two examples of pure translation. (b) Two examples of unequal translations of elements. (c) Analogs to the experimental displays. (d) Two examples of Ternus configurations. (e) Symmetric motion correspondence solutions.

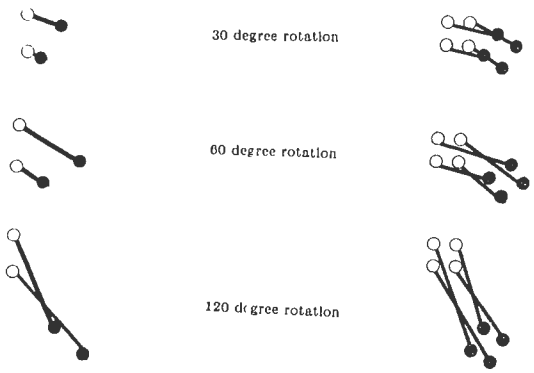


Figure 5. Performance of the model for rotations in the plane.

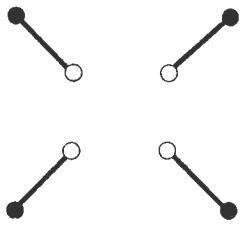


Figure 6. Performance of the model for a looming display. This solution is generated when diagonal Frame I elements are not neighbours, and when diagonal Frame I elements are neigbours.

DEVICE REPRESENTATION USING INSTANTIATION RULES AND STRUCTURAL TEMPLATES†

Mingruey R. Taie, Sargur N. Srihari, James Geller and Stuart C. Shapiro

Department of Computer Science
State University of New York at Buffalo
Buffalo, NY 14260, USA
taiemr%buffalo@csnet-relay

Abstract — A device representation scheme for automatic electronic device fault diagnosis is described. Structural and functional descriptions of devices (which are central to design-model-based fault diagnosis) are represented as instantiation rules and structural templates in a semantic network. Device structure is represented hierarchically to reflect the design model of most devices in the domain. Each object of the device hierarchy has the form of a module. Instead of representing all objects explicitly, an expandable component library is maintained, and objects are instantiated only when needed. The component library consists of descriptions of component *types* used to construct devices at all hierarchical levels. Each component *type* is represented as an instantiation rule and a structural template. The instantiation rule is used to instantiate an object of the component *type* as a module with I/O ports and associated functional descriptions. Functional description is represented as procedural attachments to the semantic network; this allows the simulation of the behavior of objects. Structural templates describe sub-parts and wire connections at the next lower hierarchical level of the component *type*. Advantages of the representation scheme are compactness and reasoning efficiency.

INTRODUCTION

First Generation diagnostic expert systems, such as MYCIN [10] for medical diagnosis and CRIB [5] for computer hardware fault diagnosis, are built on empirical rules that associate observed symptoms with possible fault (disease) hypotheses. While these systems are considered successful, experience has shown significant drawbacks in their design methodology: knowledge acquisition from domain experts is difficult; all possible faults (diseases) have to be enumerated explicitly, which results in limited diagnostic power; and they have almost no capability of system generalization.

Structural and functional descriptions, usually referred to as "design models" of a device, have been suggested as a solution to the difficulties of empirical-rule-based diagnosis systems in knowledge acquisition, diagnosis capability, and system generalization [1,2,4]. Such systems are referred to as "design-model-based" or "specification-based" as opposed to first generation systems which are "symptom-based"[6]. Diagnostic architectures for combining symptom-based and specification-based reasoning have also been proposed [11].

† This work was supported in part by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441 5700, and the Air Force Office of Scientific Research, Bolling AFB DC 20332 under contract No. F30602-85-C-0008.

The present work focuses on knowledge representation for design-model-based diagnosis. The knowledge needed for building such a system is well-structured and readily available at the time when a device is designed. There is no need to explicitly enumerate all possible faults since they are defined generically as violated expectations at the output ports. This approach makes adaptation of the system to a new device much easier, because all that is needed is to describe the device to the system.

Since a design-model-based fault diagnosis system reasons directly on the structure and function of a device and usually uses a simple inference engine, the representation of the device is vital to system performance. We use a hierarchical representation of knowledge to provide abstraction levels of devices. This allows a fault diagnosis system to focus on either individual objects or on several objects at a time.

Compactness of device representation is desirable for memory economy and diagnostic reasoning efficiency. It is observed that many parts of an electronic device often have the same component type and thus show the same function. Therefore we find that representing every detail of a device creates unnecessary redundancy. Instead of representing all objects explicitly, an expandable component library is maintained, and objects are instantiated only as needed. An object, which may be the device itself or a sub-part of it at any hierarchical level, is represented as a module.

The component library consists of descriptions of all component *types* used to construct the devices at all abstraction levels. Each component *type* is in turn abstracted at two levels: at level-1, it is a module (a black box in the usual sense) with I/O ports and functional descriptors; at level-2, sub-parts and wire connections are envisioned. In a previous implementation, two instantiation rules were used for the representation [9], this was satisfactory for simple cases, but performance degraded when dealing with more complex devices. In this paper, we present a new device representation scheme that uses both instantiation rules and structural templates in a semantic network. Functional description is represented as a procedural attachment to the semantic network. This allows the simulation of the behavior of objects.

The representation scheme has been used to represent several devices, including several multiplier/adder boards and a six channel PCM (Pulse Code Modulation) board for telephone communication, in a Versatile Maintenance Expert System (VMES) [9]. The result shows that the representation scheme is effective, and that SNePS [7], the semantic network processing system used as an underlying representation tool and inference package, is suitable for this purpose.

In the following sections, details of the representation scheme are described, an example of using the representation scheme for electronic circuit board trouble-shooting is presented, and the method of "lazy instantiation" is investigated.

REPRESENTATION SCHEME

To build a design-model-based fault diagnosis system, it is necessary to extract structural and functional information from the design model of the device. This information has to be represented in an appropriate formalism. One way to represent the device is to describe every detail of the device directly to the system. This could lead to inefficiencies in memory usage and in system development. Instead of hand-coding every detail of the device, VMES keeps a component library which describes every "*type*" of component.

The representation scheme is implemented as a semantic network for several reasons. The semantic network representation has long been around as a knowledge representation technique for expert systems [3]. It is able to represent subset and element taxonomic information, and has the potential for a smooth interface with natural language subsystems [3]. Second, a printed circuit board can be viewed as a constrained network, and it is very natural to represent it as a semantic network. Third, SNePS provides mechanisms for representing both declarative and procedural knowledge; the former is good for representing device structure, and the latter for device function.

In the representation scheme, each component type is abstracted at two levels and represented by a SNePS rule and a SNePS assertion. The former is categorized as an "instantiation rule", and the latter a "structural template". The structural representation reflects the part hierarchy of a device. Sub-parts of a device are instantiated only when they are needed. This increases memory efficiency.

Level-1 Abstraction: Instantiation Rule for I/O Ports and Function

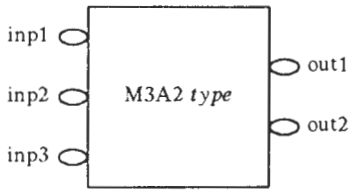
At level-1 abstraction, knowledge about a component *type* is represented as a SNePS rule. The rule is used later on to instantiate an object of the component *type* as a module with its own I/O ports and associated functional descriptor. The functional descriptor contains information about the functional description of the component *type*. The representation of the level-1 abstraction of component *type* "M3A2" is shown in Figure 1. (M3A2 is an artificial board which consists of three multipliers and two adders.) Its structure is shown in Figure 2.

Figure 1(a) shows the level-1 abstraction of the M3A2 *type*. The function of the component is abstracted as mathematical equations. This is good for digital circuits in general. Figure 1(b) and 1(c) contain our representation for the abstraction.

The first three lines of the instantiation rule shown on Figure 1(b) say that "if x is an M3A2 and is to be instantiated at level-1 abstraction (TBI-L1A), then do the following". The next five lines will instantiate the I/O ports of the object when this rule is fired. I/O ports of an object are the places where the input/output values of the object are stored. Measured (observed) I/O values depict the real behavior of the object, and calculated I/O values show its expected (normal) behavior. The last two "builds" create the functional descriptors of the object. The function of an object in the domain can be best abstracted as the relation between its inputs and outputs. The first one says "in order to simulate the value of the first output, use the function M3A2out1 which takes three parameters namely the inputs of the object x in order". Similar functional descriptors can be included for the input ports if the inference of input value from outputs and other inputs is desired (these are not shown in the figure).

The functional description should be usable to simulate the component behavior, i.e., to calculate the values of output ports if the values of the input ports are given. It should also be usable to infer the values of the input ports in terms of the values of other I/O ports. This is important if hypothetical reasoning is used for fault diagnosis. Though at this stage, VMES only uses the functional description to calculate values at output ports, our representation scheme can be used both ways.

As shown in Figure 1(b), the functional descriptor of a port contains a pointer to its functional description as well as other information concerning the use of the functional description. The functional description itself is implemented as a LISP function (see Figure 1(c)), which calculates the desired port value in terms of the values of other ports. Every port of a component type has such a function associated with it. Some more discussion about functional representation is given in Section 4..



out1 = inp1*inp2 + inp1*inp3
out2 = inp1*inp3 + inp2*inp3

Figure 1(a). Level-1 abstraction of component *type* M3A2.

```
(build
   avb $x
   ant (build object *x type M3A2 state TBI-L1A)
   cq  ((build inport-of *x inp-id 1) = INP1
        (build inport-of *x inp-id 2) = INP2
        (build inport-of *x inp-id 3) = INP3
        (build outport-of *x out-id 1) = OUT1
        (build outport-of *x out-id 2) = OUT2
        (build port *OUT1 f-rule M3A2out1
               pn 3 p1 *INP1 p2 *INP2 p3 *INP3)
        (build port *OUT2 f-rule M3A2out2
               pn 3 p1 *INP1 p2 *INP2 p3 *INP3]
```

Figure 1(b). Instantiation rule for the level-1 abstraction of component *type* M3A2: I/O ports and functional descriptors. Variables are shown in italics, and "*" is a SNePS macro for variable value substitution.

```
(defun M3A2out1 (inp1 inp2 inp3)
   (plus (product inp1 inp2)
         (product inp1 inp3]

(defun M3A2out2 (inp1 inp2 inp3)
   (plus (product inp1 inp3)
         (product inp2 inp3]
```

Figure 1(c). Functional description of component *type* M3A2.

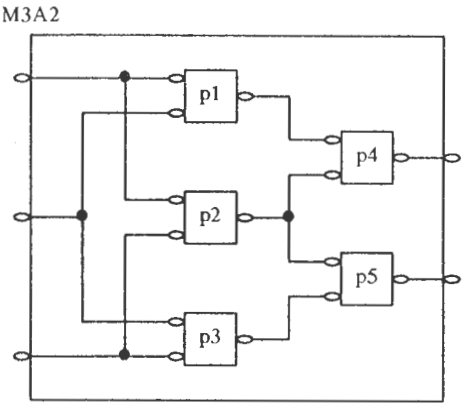
Level-2 Abstraction: Structural Template for Subparts and Wire Connections

At the level-2 abstraction, a structural template, which is implemented as a SNePS assertion, is used to describe the sub-parts of the object at the next hierarchical level, and the wire connections between the object and its sub-parts, as well as those among the sub-parts themselves. In figure 2(a), the abstraction of component *type* M3A2 at this level is illustrated. Note that the sub-parts are abstracted at their own level-1 abstraction, i.e., modeled as modules with I/O ports. The component *types* of sub-parts are also indicated.

The structural template representation is shown in Figure 2(b). The representation can be viewed as consisting of three parts. The first part, which is the second line of Figure 2(b), denotes that the representation is the structural template (ST) for component *type* M3A2 at level-2 abstraction (L2A). The second part describes the sub-parts. Associated with each sub-part are a part-id, an ext-name, and a class indicator. The part-id identifies the sub-part of the component *type*. The ext-name is for name extension, and class is the component *type* of the sub-part. This information is used for instantiating a sub-part. For example, if when diagnosing a device D1 of *type* M3A2, the second sub-part (with part-id M3A2-p2 inside the structural template) is found suspicious, then an object is created with a name of D1-M2 and a type of MULT. The last part of the structural template specifies the wire connections shown in Figure 2(a).

A structural template provides the necessary knowledge about the sub-structure of all objects of the same component *type* without representation overhead. Unlike instantiation rules, structural templates are never executed (fired) to produce a representation for any specific object. When reasoning on the sub-structure of an object is required, instead of instantiating the sub-structure (all the sub-parts and wire connections) and then reasoning on the resulted representation, we do it directly on the structural template of the object. If suspicious sub-parts are located, they (but not all sub-parts) are instantiated at the level-1 abstraction by the instantiation rules for further examination.

Device representation by instantiation rule and structural template is very compact and effective. In the next section, an application example of using this representation scheme is demonstrated.



Multiplier: p1, p2, p3
Adder: p4, p5

Figure 2(a). Level-2 abstraction of component *type* M3A2.

```
(build
  type M3A2 state ST-L2A
  sub-parts
    ((build part-id M3A2-p1 ext-name M1 class MULT)
     (build part-id M3A2-p2 ext-name M2 class MULT)
     (build part-id M3A2-p3 ext-name M3 class MULT)
     (build part-id M3A2-p4 ext-name A1 class ADDER)
     (build part-id M3A2-p5 ext-name A2 class ADDER))
  connections
    ((build from (build inport-of M3A2 inp-id 1)
          to   ((build inport-of M3A2-p1 inp-id 1)
                (build inport-of M3A2-p2 inp-id 1)))
     (build from (build inport-of M3A2 inp-id 2)
          to   ((build inport-of M3A2-p1 inp-id 2)
                (build inport-of M3A2-p3 inp-id 1)))
     (build from (build inport-of M3A2 inp-id 3)
          to   ((build inport-of M3A2-p2 inp-id 2)
                (build inport-of M3A2-p3 inp-id 2)))
     (build from (build outport-of M3A2-p1 out-id 1)
          to   (build inport-of M3A2-p4 inp-id 1))
     (build from (build outport-of M3A2-p2 out-id 1)
          to   ((build inport-of M3A2-p4 inp-id 2)
                (build inport-of M3A2-p5 inp-id 1)))
     (build from (build outport-of M3A2-p3 out-id 1)
          to   (build inport-of M3A2-p5 inp-id 2))
     (build from (build outport-of M3A2-p4 out-id 1)
          to   (build outport-of M3A2 out-id 1))
     (build from (build outport-of M3A2-p5 out-id 1)
          to   (build outport-of M3A2 out-id 2))]
```

Figure 2(b). Structural template for the level-2 abstraction of component *type* M3A2: sub-parts and wire connections.

AN APPLICATION EXAMPLE

We are developing a versatile maintenance expert system (VMES) for digital circuit trouble-shooting [8]. VMES is intended to be versatile in several senses: good for a wide range of devices in the domain; good for most common faults in the domain; and able to communicate with the user by several media [9]. VMES consists of two major modules: an expandable component library for device representation and an inference engine for diagnostic reasoning. The representation scheme described above is used for the current implementation of the component library.

Inference Engine of VMES

The inference engine for fault diagnosis follows a simple control structure. It starts from the top level of the structural hierarchy of the device and tries to find output ports that violate an expectation. "Violated expectation" is defined as a mismatch between the expected (calculated) value and the observed (measured) value at some output. After detecting a violated expectation, the system reasons on the structural template to find a subset of components at the next lower hierarchical level which might be responsible for the bad outputs. This process is then continued with the suspicious parts. A part is declared faulty if it shows some violated expectation at its output port and it is at the bottom level of the structural hierarchy. The bottom of the hierarchy will contain the smallest replaceable units for the intended maintenance level. In other words, if a device can be replaced but not repaired in a certain situation, then there is no need to represent its internal structure.

The inference engine is a rule-based system implemented in SNePS. The control flow is enforced by a LISP driving function called "diagnose". SNePS can do both forward and backward inference, and is capable of doing its own reasoning to diagnose a fault. The LISP driving function has been introduced for efficiency reasons only.

A small set of SNePS rules is activated at every stage of the diagnosis. For example, three rules are activated when reasoning about a possible violated expectation of a specific port of a device. One rule is to deduce the measured value of the port. If the value can not be deduced from the wire connections, the rule would activate a LISP function which asks the user to supply one. A similar rule is activated for the calculated value, and the last rule is used to compare the two values to decide if there is a violated expectation. Figure 3 shows the last rule in both SNePS code and in English.

In SNePS code:

```
(build
  avb ($p $vc $vm)
  &ant ((build port *p value *vc  source calculated)
        (build port *p value *vm source measured))
  cq (build
        min 1 max 1
        arg (build name: THEY-MATCH p1 *vc p2 *vm)
        arg (build port *p state vio-expct]
```

In English:

If the calculated and measured values of port p are vc & vm, one and only one of the following statements is true:
(1) vc and vm agree;
(2) port p displays a violated expectation.

Figure 3. SNePS rule for detecting violated expectation at output ports.

The diagnosis strategy along with the combination of a LISP driving function and SNePS rules turns out to be very efficient. The diagnosis can be monitored by the SNePS text or graphic inference trace.

A Diagnostic Example

Figure 4 shows the representation scheme used by VMES in diagnosing a multiplier/adder board. Again, the component *type* M3A2 is used as our example.

We first name the board D1. Figure 4(a) shows the result of instantiating D1 using the instantiation rule for M3A2 *type* device (see Figure 1(b)). After the instantiation, D1 has its own I/O ports and functional descriptors, and thus its I/O values can be assigned. The result of value assignments is also shown in Figure 4(a). Then the inference engine begins to check the outputs of D1 by using the functional description of D1. It concludes that there is a violated-expectation at the first output port of D1 as shown in Figure 4(b), since the expected value, which is calculated using the functional description, should be a "4" instead of the observed "2".

At this stage, it is necessary to check the substructure of D1 to locate the faulty parts. Thus VMES turns to the structural template for M3A2 (see Figure 1(c)), i.e., the component *type* of D1. From the wire connection depicted by the structural template, VMES determines that sub-parts p1, p2, and p4 of D1 may be responsible for the malfunctioning of D1. This is shown in Figure 4(c). Note that sub-part p2 may be excluded if a "single fault assumption (SFA)" is made for diagnosis. The reason is that a faulty sub-part p2 is inconsistent with the observed behavior of D1 under SFA. In other words, a bad output of sub-part p2 should cause violated-expectation at both output ports of D1, but this is not the case.

Suppose SFA is not made for this example. The next step is to instantiate all suspicious sub-parts of D1, and move the diagnosis process to those sub-parts. Figure 4(d) shows the instantiation of these sub-parts. Sub-part p1 is instantiated as D1-M1, p2 as D1-M2, and p4 as D1-A1 using the information supplied by the structural template of the component *type* of D1, i.e., M3A2. Note that sub-parts p3 and p5 are not touched at all. This is the main advantage of this representation scheme.

Now the diagnostic reasoning process moves to D1-M1, D1-M2, and D1-A1 with the same inference strategies used for diagnosing D1. As shown in Figure 4(e), D1-M1 and D1-M2 show no problem, but D1-A1 shows a violated expectation at its output. The process will turn to the structural template of the component *type* of D1-A1 (an ADDER) if D1-A1 is not an object at the bottom level of the structural hierarchy. This is not the case in this simple example, where D1-A1 is an SRU (smallest replaceable unit) for the intended maintenance level. Therefore, D1-A1 is finally identified as the faulty part.

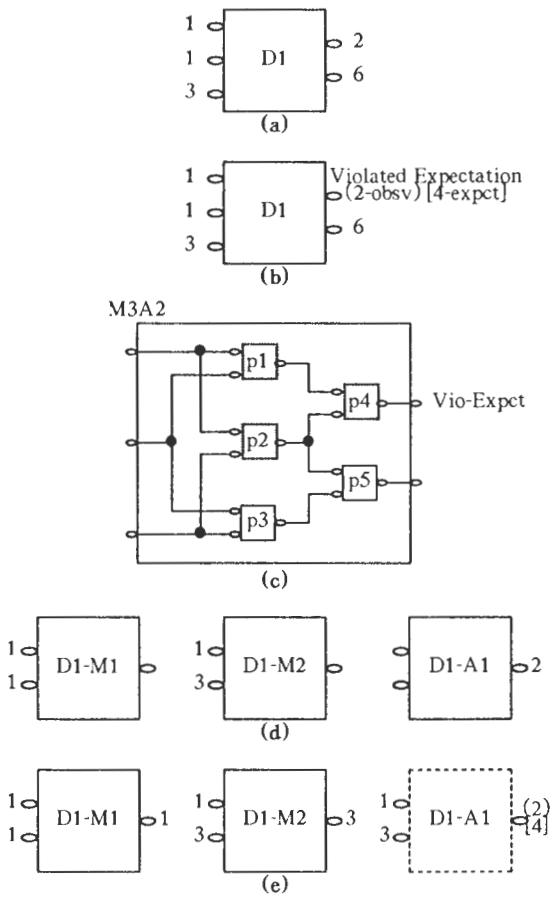


Figure 4. A diagnostic example for the device representation scheme. (D1 is an M3A2 *type* device).

DISCUSSION

Hand coding every detail, i.e., all sub-parts at all hierarchical levels, of a device is inefficient. It results in unnecessary redundancies in device representation since many parts of a device may be identical in the domain of electronic circuit boards. For instance, the six PCM chips on a six channel PCM board are exactly the same.

Representing a device as a hierarchically arranged set of objects, each of which is modeled as a module, is hardly a new idea. What is significant in our representation scheme are the clear distinction between the two levels of the abstraction and the use of an instantiation rule and a structural template to represent the different levels. The representation scheme along with an expandable component library leads to several important advantages: compact representation and system efficiencies in both system development and operating phases.

We first claim that a clear distinction between the two level abstractions of an object is desirable. In some points during diagnosis, we would like to treat an object as a complete black box — that means only the knowledge from the level-1 abstraction, which consists of I/O ports and a functional description of the object, is needed. To represent the sub-structure of the object, which is the level-2 abstraction, together with the level-1 abstraction is inefficient, since the sub-structure of the object may never be needed.

The use of structural templates to represent the substructure of objects of a component *type* has advantages over a procedural representation which uses a procedure or an instantiation rule for it [1, 9]. Whenever it is needed to reason about the substructure of an object, it is carried out on the unique structural template for the component *type* of the object. Only the subparts that requires further examination will be instantiated (by the proper instantiation rules for them). Unlike the structural template representation, a procedural representation is used to instantiate "all" sub-parts of an object, and then the reasoning is carried out over the resulting substructures. This creates unnecessary representation, and thus is memory inefficient. This is also execution inefficient due to the overhead of instantiating all sub-parts. An extreme example is an object with one hundred sub-parts at the next hierarchical level, only three of them needing further investigation. The advantage of the structural template is quite significant in memory and processor critical environments, such as the widespread microprocessor-based computers.

Since different ports of different component *types* might have the same function, some functions can be shared. For instance, the simple function "ECHO" defined as:

(defun ECHO (inp) inp)

is shared by several different component *types* namely by the type "super-buffer", the type "wire" and the type "1-to 1 transformer". All these component *types* show the same behavior at our level of component abstraction: they echo the input to the output. As depicted above, the functional description is versatile in that it supports the simulation and the inference of the device behavior; it also supports hypothetical reasoning and the representation scheme is quite simple.

Along with the representation scheme using instantiation rules and structural templates is the idea of an expandable component library. This makes life very easy in adapting VMES to other devices. All that the user has to do is to add the structural and functional information of the "new" component *types* to the component library. A new component *type* is defined as a component *type* which has not been described to the component library. The new device itself is a new component type by our definition. The effort required to adapt the system to new devices should be minimal since digital circuit devices have a lot of common components, and the structural and functional description are readily available at the time a device is designed.

In order to test this idea as well as the suitability of hierarchical structural representation, we invented another artificial device *type* called XM3A2 and entered its description into the system. The XM3A2 *type* has three inputs and two outputs exactly like the M3A2 *type*, but it only has a single subpart which is of M3A2 *type*. Actually, it is a device which has an extra layer of packaging on top of an M3A2 *type* device. Given that the M3A2 *type* has been known to the system, only the XM3A2 *type* had to be added, which was done by adding a simple instantiation rule and a simple structural template. There was no need for a new functional description since the function of XM3A2 is the same as the function of M3A2. The XM3A2 device has three levels of structural hierarchy, and our test successfully found the faulty part at the lowest level. Though the example of XM3A2 is somewhat simplistic, it shows the capability of our system to deal with a wide range of devices in the domain with arbitrary complexity. Actually, a real six channel PCM board has been represented and a fault has been successfully located.

Acknowledgement

The authors would like to thank Scott S. Campbell, Dale Richards and Norman Sturdevant for their help and useful comments on the development of VMES.

References

1. R. Davis and H. Shrobe, "Representing Structure and Behavior of Digital Hardware," *Computer*, pp. 75-82 (Oct. 1983).
2. R. Davis, "Diagnostic Reasoning Based on Structure and Behavior," *Artificial Intelligence* **24** pp. 347-410 (1984).
3. R. O. Duda, P. E. Hart, N. J. Nilson, and G. L. Sutherland, "Semantic Network representations in Rule-Based Inference Systems," in *Pattern-Directed Inference Systems*, ed. F. Hayes-Roth,Academic Press, New York (1978).
4. M. R. Genesereth, "The Use of Design Descriptions in Automated Diagnosis," *Artificial Intelligence* **24** pp. 411-436 (1984).
5. R. T. Hartley, "How Expert Should an Expert System Be?," pp. 862-867 in *Proceedings of the 7th International Joint Conference on AI*, (August 1981).
6. J. J. Richardson, R. A. Keller, R. A. Maxion, P. G. Polson, and K. A. DeJong, "Artificial Intelligence in Maintenance: Synthesis of Technical Issues," AFHRL-TR-85-7, Airforce Systems Command, Air Force Human Resources Laboratory, Brooks Air Force Base, TX 78235 (1985).
7. S. C. Shapiro, "The SNePS Semantic Network Processing System," pp. 179-203 in *Associative Networks: The Representation and Use of Knowledge by Computers*, ed. Nicholas V. Findler,Academic Press, New York (1979).
8. S. C. Shapiro, S. N. Srihari, J. Geller, and M. R. Taie, "A Fault Diagnosis System Based on an Integrated Knowledge Base," *IEEE Software* 3(2)(March, 1986).
9. S. C. Shapiro, S. N. Srihari, M. R. Taie, and J. Geller, "VMES: A Network-Based Versatile Maintenance Expert System," pp. (to appear) in *Proc. of 1st International Conference on Applications of AI to Engineering Problems*, , Southampton, U.K. (April 1986).
10. E. H. Shortliffe, *Computer-Based Medical Consultations: MYCIN*, American Elsevier/North Holland, New York (1976).
11. Z. Xiang and S. N. Srihari, "A Strategy for Diagnosis Based on Empirical and Model Knowledge," pp. (to appear) in *Proc. of Sixth Int. Workshop on Expert Systems and their Applications*, , Avignon, France (April 1986).

MACHINE TRANSLATION BETWEEN CHINESE AND ENGLISH

Wanying Jin
Dept of Computer Science
University of Texas at Austin
Austin, Texas 78712, U. S. A.

ABSTRACT

The paper describes the strategy and implementation of symmetric translation of Chinese and English. The system consists of two sets of grammar rules and a set of translation rules as axioms in the form of procedural logic. Implementation is accomplished by HCPRVR --- a LISP version of PROLOG. The strategy of the system includes three steps: First, the source language is parsed into Semantic Relation SR, then translation is accomplished between Chinese Semantic Relation SRc and English Semantic Relation SRe, and finally the target language is generated from SR. The translation mechanism is discussed in detail. The program has been tested successfully on a limited set of sentences.

1. Introduction:

A symmetric procedural logic grammar was investigated by Simmons and Chester [1] to accomplish both parsing from English to logical form and generating English from the logical form. Later, some experimental grammars were developed to translate English to Japanese [2] or to translate Horn clauses from English [3], [4]. The problem remaining in this research is that the translation rules work only in one direction. Usui questioned whether it is possible to develop symmetric paraphrase rules to accomplish bi-directional translation. The attempt of this research is to develop symmetric rules for bi-directional English and Chinese translation. This experiment demonstrates that symmetric ruleform can be designed to achieve symmetric translation at least for fragments of two languages.

The study was conducted on a small set of sentences requiring a limited set of grammar rules and translation rules. Implementation is embodied by HCPRVR [5] which is also documented [6]. The source language is first parsed into Semantic Relation SR, then SR of one language is translated into SR of another language. Finally, the target language is generated from SR. The entire procedure is described as follows:

English <==> SRe <==> SRc <==> Chinese

By using symmetric grammar rules, the translation can be made in either direction. As Chinese is quite different from English , it is usually impossible to get a good quality translation in the surface language level by word-for-word translation. However, both languages can be analyzed into deep case structures which have case structure names common to English and Chinese (e.g. AGenT, Affected-Entity TIME, LOCation etc). Each language has some constituents which may not exist in the other language. Therefore, a set of translation rules that paraphrase case structures from one language to those of the other is essential.

If symmetry is maintained in the translation rules, then symmetric translation using only one set of translation rules is possible. [7]

2. Difference between English sentences and Chinese sentences:

2.1. Word order:

Word order in a Chinese sentence is quite different from that in English. Consider two sentences:[1]

Since 1980 the Chinese puppet troupes have made several visits to Europe and the United States.
Zicong 1980 nian yilai Zhongguo muou jutuan yijing dao Ouzhou he Meiguo jinxingle duoci fangwen.
*(Since-pre 1980 year since-post Chinese puppet troupes have to Europe and the United States made several visits.)

English grammar:
S1 = PP NP AUX VERB NP1 PP
Chinese grammar
S2 = PP NP1 AUX PP VERB NP1

The author is on leave from Shanghai University of Technology, Shanghai, P. R. China.

[1] in order to make the Chinese sentences readable, each Chinese sentence is followed by a word-for-word English version in the parentheses marked by * .

In this example the Chinese PP is placed between AUX and VERB, while in English PP follows the verb and its object.

2.2. The constituent ordering of PP:

A Chinese PP has several different constituents:

PP::= PREP NP PREP | PREP NP | NP PREP

In the example above the preposition *since* is split into two parts: *zicong* and *yilai*. The year *1980 nian* is put in between.

since 1980 ----> zicong 1980 nian yilai

*(since-pre 1980 year since-post)

The following example shows the order difference between English PPe and Chinese PPc:

The visit of Chinese puppet troupe

NPe PPe = PREP NPe

Zhongguo muou jutuan de fangwen

PPc = NPc1 PREP NPc

*(Chinese puppet troupe of visit)

The preposition *(of) follows NPc1 *(Chinese puppet troupe) and the preposition phrase *(Chinese puppet troupe of) is put in the front of NPc *(visit).

2.3. Determiner:

In Chinese sentences, the determiner *the* is not used. In translating any English sentence into a Chinese sentence, a rule simply deletes the determiner *the*, but in translating from Chinese into English two questions arise:

In which cases should a noun phrase use the determiner **the** *?*

Inside the case structure SRe, where should the constituent pair <DET the> be placed?

2.4. English idiomatic expression interpretation:

They brought this Chinese ancient art form to the attention of more foreigners.

The English idiomatic expression *to bring something to somebody's attention* first must be interpreted as *let somebody notice something* or *let something be noticed by somebody*, then be translated into Chinese. The Chinese sentence will look like:

Tamen shide zhezhong Zhongguo gudaide yishu xingshi yinggile henduo waiguoren de zhuyi.

*(they let this Chinese ancient art form cause more foreigners of attention.)

2.5. Date representation:

In Chinese, *day, month* or *year* must follow the date and describe the year first, month next and day last.

in October 1984 ---> zai 1980 nain 10 yue

*(in 1980 year 10 month)

2.6. Multiple interpretation:

Frequently, one English word with several meanings corresponds to several Chinese words. For example, in this study the preposition "to" has two interpretations:

to the attention --- *TO

to Europe --- LOCation

. The following four sentences were chosen to illustrate the solutions to the problems discussed above.

- *Since 1980 Chinese puppet troupes have made several visits to Europe and the United States.*
- *They brought this Chinese ancient art form to the attention of more foreigners.*
- *The 1981 American debut of the Fujian puppet troupe was an unparalleled success.*
- *In October 1984 the visit of the Shanghai Rod puppet troupe delighted San-Francisco audiences.*

3. Mechanism of translation:

This project consists of three sets of axioms (MY-SRc, MY-SRe, and MY-TRANS), vocabulary entity (MY-WORDS) and semantic event forms (MY-SEF): [6]

MY-SRc --- a set of Chinese grammar rules Sc,

MY-SRe --- a set of English grammar rules Se,

MY-TRANS --- a set of translation rules TRANS0,

MY-WORDS --- a vocabulary entity, and

MY-SEF --- a set of semantic event forms.

The basic rule-forms are outlined below:

Se --- parse English sentence into English semantic relation SRe and vice versa.

(Se <English sentence> <SRe>)

Sc --- parse Chinese sentence into Chinese semantic relation SRc and vice versa.

(Sc <Chinese sentence> <SRc>)

TRANS0 --- translate SRe into SRc and vice versa.

(TRANS0 <SRe> <SRc>)

Vocabulary entity is a set of tuples with 4 or 5 items:

(constituent English word-feature Chinese)

(constituent English word-feature Chinese constraint)

Semantic event form is a set of triples which constrains the semantic relation between the phrases.

(phrase-feature phrase-feature semantic-relation)

3.1. Grammar rules MY-SRe:

A set of context-free grammar rules MY-SRe is designed to parse English sentences, and to order the constituents under the control of semantic constraints.

```
Se   ::= NPe VPe | PPe NPe VPe
NPe  ::= ART NPe1 PPe | ART NPe1 | NPe1
NPe1::= ADJ NPe1 | NOUN | PRON
VPe  ::= AUX VERB VCOMPe | VERB VCOMPe
PPe  ::= PREP NOUN NOUN |
         PREP NPe *AND NPe | PREP NPe
VCOMPe::= NPe PPe | NPe | NIL
```

The grammar rules are written in the form of procedure logic. For example:

```
((Se X (W U X1 .W1)) <
                (NPe X NF Y X1)
                (VPe Y VF NIL (W .W1))
                (NF VF U))
```

Semantic event form (NF VF U) indicates that the semantic relation between NP and VP must match against U.

3.2. Grammar rules MY-SRc:

For Chinese sentences, grammar rules MY-SRc reflect the ordering of Chinese constituents.

```
Sc  ::= NPc VPc | PPc NPc VPc
NPc ::= ART NPc1 | PPc NPc | NPc1
```

NPc1::= ADJ NPc1 | NOUN | PRON
VPc ::= VERB VCOMPc | AUX NPc VERB NPc
| AUX PPc VPc | PPc VPc
PPc ::= PREP NOUN | PREP NOUN NOUN
| PREP NPc NPc | NPc PREP
VCOMPc::= PPc NPc | NPc | NIL

The procedural logic forms are written according to the Chinese grammar rules. An example for rule Sc ::= PPc NPc VPc is shown below:

((Sc X (W U Y1 S X1 .W1)) <
(PPc X PF Y X1)
(NPc Y NF Z Y1)
(VPc Z VF NIL (W .W1))
(VF PF S)(NF VF U))

The grammar rules for English and Chinese are designed to provide an ordering of the case analysis to minimize the translation rules.

3.3. Translation rules MY-TRANS:

Based on conventional differences between the two languages, the context-sensitive translation rules substitute, delete and add constituent pairs from the target lexicon for the source lexicon to achieve the symmetric translation. The following two examples show how the rules implicitly map the word orderings to accomplish better translation:

3.3.1. Translating noun phrase:

The English noun phrase (*the visit of Chinese puppet troupe*) matches against the grammar rule NPe = ART NPe1 PPe, while the Chinese noun phrase (*Zhongguo muou jutuan de fangwen*) *(Chinese puppet troupe of visit) matches the grammar rule NPc = PPc NPc. A translation rule changes the word ordering according to both grammar rules, e.g.:

ART NPe1 PPe <==> PPc NPc

3.3.2. Translating verb phrase:

In the same way two grammar rules:

VPe = AUX VERB NPe PPe
VPc = AUX PPc VERB NPc

contrast the English verb phrase VPe, (*have made several visits to Europe and the United States*) with the Chinese verb phrase VPc, (*yijing dao Ouzhou he Meiguo jinxingle duoci fangwen*) *(have to Europe and the United States made several visits). Reordering

AUX VERB NPe PPe <==> AUX PPe VERB NPe

must be accomplished during the process of translation. In this study in order to minimize the translation rules, reordering is made in grammar rules before/after translation to achieve the most efficient translation.

3.3.3. Substituting constituents:

Substituting target language equivalents from the vocabulary entity, the translating rules MY-TRANS also resolve the remaining problems mentioned in section Two.

Rule

((TRANS (W (X DET the . S) .R) (W (X1 .S1) .R1)) <
(NOUN X NF X1 the)
(TRANS S S1)(TRANS R R1))

will delete the semantic relation pair (DET the) when translating English into Chinese and will add it to the right place when translating Chinese into English. In the same way, rule

((TRANS (TIME (X .S) .R)
(TIME (X1 MSR X2 .S1) .R1)) <
(NOUN X TIME X1 X2)
(TRANS S S1)(TRANS R R1))

handles the date representation when the noun has feature TIME and constraint **nian.**

Rule

((TRANS (PREP X .R) (PREP X1 PREP X2 .R1)) <
(PREP X PF X1 X2)
(TRANS R R1))

will substitute X1, X2 for X when translating the English preposition *since* into the Chinese preposition *zicong, yiilai* and vice versa.

3.4. Vocabulary entity MY-WORDS and semantic event form MY-SEF:

Both English and Chinese grammar rules share the same vocabulary entity. Some vocabulary tuples have <constraint> which indicates some extra word needs to be added when this word is matched. For example: (noun 1980 TIME 1980 nian) indicated that when noun 1980 is interpreted as time, the extra word *nian* *(year) must be added.

A Semantic Event Form (SEF) is a set of triples which describes the semantic constraints between the constituents. For example, (VF NF S) indicates the feature VF of verb phrase and the feature NF of noun phrase are constrained by semantic relation S. If (VF NF S) is bound to (ACT LOC LOC), then the semantic relation between verb and its object must be LOCation provided verb has feature ACTion and noun phrase has feature LOCation. A set of semantic event forms constrains the translation to semantically well-formed constituents.

In parsing surface sentences, multiple interpretation is controlled by a set of semantic event forms. For example, (ACT LOC LOC) will constrain *to* to be interpreted as *dao* because the preposition feature LOC matched. In contrast (ACT ACTION *TO) will constrain *to* to be kept as the same (in the case of *to the attention*).

The axioms MY-SRe, MY-SRc, MY-TRANS, MY-WORDS and MY-SEF complete the translation mechanism.

4. Conclusion:

This program has been tested on a limited set of sentences. The problems have been solved fairly well by the rule forms described. Translating the four English sentences to Chinese and the four Chinese sentences to English gives the following results shown in table 1. The grammar gave multiple interpretations for each sentence, ranging from 2 to 8 analyses each. Two interpretations were judged to be poor because of the incorrect word ordering.

sentence	number of interpretations	number of sentence types	quality justification
E1	4	2	4 good
E2	4	1	4 good
E3	2	1	2 good
E4	6	3	4 good 2 poor
C1	4	1	4 good
C2	4	1	4 good
C3	4	1	4 good
C4	8	2	8 good
8	36	12	34 good 2 poor

The grammar was then rewritten to achieve fewer interpretations. The total number of interpretations was reduced to 22 and resulted in good translation. As the size of the grammar and vocabulary increase, the translating speed obviously slows down and subtle errors may result in poor translation.

We conclude that: *it is definitely possible to write grammars to translate between two subset of languages using symmetric rule forms.* [7]

5. Discussion:

5.1. Balance between efficiency and generality:

Writing grammar rules generally or recursively will make the translation more flexible. However, it might cause poor-quality translation if the constraints are inappropriate. Also it will slow down the translating process.

Associating the general syntactic rules with semantic event forms reduces the number of possible interpretations by rejecting those syntactic interpretations which are not covered by semantic forms. For some special cases writing grammar rules with sharp semantic constraints increases the translating accuracy. For example, writing rule PP = PREP NOUN NOUN specially for date representation as "in October 1984" and constraining the interpretation with a semantic event form to the classes (IN/AT TIME TIME) will be more efficient than the general rule PP = PREP NP NP | PREP NP NCOMP.

This study keeps the balance between generality and efficiency by means of writing general rules associated with SEFs and purposely writing special rules for some special phrases.

5.2. Rule order affects the efficiency and accuracy of translation:

For English grammar:

Se ::= NPe VPe | PPe NPe VPe
PPe::= PREP NOUN NOUN
| PREP NPe *AND NPe
| PREP NPe

since all PPs start with a preposition , a sentence beginning with PP will fail at the first word parsing for the rule Se = NPe VPe. A sentence beginning with NPe also will fail at the first word parsing for the rule Se ::= PPe NPe VPe. In this case, the rule order is not a problem. On the contrary, the rule order in Chinese grammar affects the efficiency.

Sc ::= PPc NPc VPc | NPc VPc
PPc::= PREP NOUN NOUN | PREP NOUN
| PREP NPc NPc | NPc PREP

In the case of PPc = NPc PREP, sentence grammar will look like Sc ::= NPc PREP NPc VPc. Comparing this rule with Sc ::= NPc VPc, it is difficult to tell which grammar rule will match the sentence until reaching the preposition. If the first rule turns out to be a failure, the second rule will be tried, then the NPc will be recomputed once again which is very inefficient. One solution is to assert the NPc when parsing NPc succeeds.

The rule ordering also affects the accuracy of the translation. In the case

VCOMPc ::= NPc PPc | NPc

writing VCOMPc = NPc first will cause incorrect translation. When parsing NPc succeeds, the PPc part following NPc will be missing. Writing rules recursively

VCOMPc ::= NPc VCOMPc | PPc VCOMPc | NIL

can solve this problem, but, again, it will affect the translating speed.

The strategy used for ordering rules is:

- *If special rules are subset of general rules, write the special rules first.*
- *If special rules are not a subset of general rules, write the frequently used rules first.*

5.3. The role of Semantic Event Forms --- SEFs:

Syntactic grammar rules syntactically guarantee the translation accuracy. The SEFs are a convenient method for selecting sense meanings and appropriate case relations for nominal phrases and as a result provide semantic constraints which to some extent improve the interpretation.

During the implemention of Chinese-English translation the SEFs accomplished the following roles:

- Control the word order --- SEFs select case relation names to rearrange the word order which results in better quality translation.
- Avoid ambiguity:

  Sc = PPc NPc VPc
  (PPc modifies VPc)
  Sc = NPc VPc => PPc NPc VPc
  (PPc modifies NPc.)

  Given two sentences:

  Zai 1980 nian Zhongguo muou jutuan fangwenle Meiguo.

  *(In 1980 Chinese puppet troupe visited the U.S.).

  Zhongguo muou jutuan de fangwen doulele San-Francisco guangzhong.

  *(Chinese puppet troupe of visit delighted San Francisco audiences)

  Mapping (VF PF TIME) against (ACT TIME TIME) the first sentence contrasts with the first grammar rule, and the second sentence matches the second rule by mapping (NF PF *OF) against

(EVT HUMAN *OF). As a result the ambiguity is avoided.

- Minimizing multiple interpretations: Preposition *to* has two interpretations in the following phrases:
  "to Europe and the U.S."
  "to the attention of more foreigners"
  By assigning feature LOC to the first PP and feature *TO to the second PP, associated with (ACT LOC LOC) and (ACT ACTION *TO) the problem of different interpretations of the same preposition was solved.

6. Further research:

Some study can be further developed in deeper case structures.As natural languages are strongly dependent on the human environment, a sentence existing in one language may not exist in another language. Therefore, it is necessary to paraphrase the sentence before translating it. For example, the English sentence *he called a black and white from the 7-Eleven* needs to be paraphrased to *he called a police from grocery* before translating it to Chinese, and the Chinese idiom *he has a bamboo in his chest* should be paraphrased as *he is full of confidence* before translation. It seems to be plausible to further develop the deep structure to solve these problems. The strategy is expected as follows:
English==SRe==SRe*i*==SRc*i*==Chinese*i* or
English==SRe==SRc==SRc*i*==Chinese*i*
e.g. paraphrase can be made before/after translation. Keeping the symmetric rules in every level, the bi-directional translation can be achieved. Obviously, the power of the system will strongly depend on the linguistic knowledge and world knowledge as well.

7. Acknowledge:

The author would like to express her sincere gratitude to Dr. Robert F. Simmons for his valuable contribution of the theory of semantic relations, his insightful conclusion as well as his help of improving the English version of this paper. She thanks to the University of Texas at Austin for providing the facilities to accomplish this study. The author also acknowledges the generous support for this research received from her home university --- Shanghai University of Technology.

8. Appendix --- Detailed procedure of the translation:[7]

```
*{; 1. Translate English sentence into SRe.        }
((Se (since 1980 Chinese puppet troupes have made
      several visits to Europe and the US)
     (made AGT (troupes NAME Chinese TYPE puppet)
           TIME (1980 PREP since)
           AUX have
           LOC (Europe PREP to *AND (US DET the))
           AE (visits MODEL several))))
*{; 2. Translate SRe into SRc.                     }
((TRANS0 (made AGT   (troupes NAME Chinese
                             TYPE puppet)
               TIME  (1980 PREP since)
               AUX   have
               LOC   (Europe PREP to
                             *AND (US DET the))
               AE    (visits MODEL several))
 (jinxingle AGT  (jutuan NAME Zhongguo
                         TYPE muou)
            TIME (1980 MSR nian PREP zicong
                                PREP yilai)
            AUX yijin
            LOC (Ouzhou PREP to *HE (Meiguo))
            AE  (fangwen MODEL duoci))))
*{; 3. Translate SRc into Chinese sentence.        }
((Sc (zicong 1980 nian yilai Zhongguo muou jutuan yijin
     dao Ouzhou he Meiguo jinxingle duoci fangwen)
     (jinxingle AGT (jutuan NAME Zhongguo
                            TYPE muou)
                TIME (1980 MSR nian PREP zicong
                                    PREP yilai)
                AUX yijin
                LOC (Ouzhou PREP dao *HE (Meiguo))
                AE  (fangwen MODEL duoci))))
```

REFERENCE

1. Simmons, R. F. and Chester, D. L., Relating Sentences and Semantic Network with Clausal Logic, CACM, August, 1982.
2. Usui, T., An Experimental Grammar for Translating English to Japanese, Tech. Rep. TR-201, Dept. of Comp. Sci. University of Texas at Austin, 1982.
3. Kowalski, R. A., Logic for Problem Solving, North-Holland, Amsterdam, 1979.
4. Yu, Yeong-Ho, Translating Horn clauses from English, AI-TR-84-3, AI lab, Dpet of Computer Science, University of Texas At Austin, August 1984.
5. Chester. D., HCPRVR: A Logic Program Interpreter in LISP, proc.AAAI 1980, ms.,Dept of Computer Science, University of Texas at Austin,1980.
6. Simmons, R. F. Computations From English, Prentice-Hall, Inc. 1984.
7. Jin, Wanying., & Simmons, R. F. Symmetric Roles for Translation of English and Chinese. Tech.Rep. AI lab, Dept of Comp.Sci. The University of Texas at Austin. March 1986.

INTER-WORD CONSTRAINTS IN VISUAL WORD RECOGNITION

Jonathan J. Hull

State University of New York at Buffalo
Department of Computer Science
Buffalo, New York 14260

ABSTRACT

A method of using knowledge about constraints *between* words in a technique for reading a running text is presented. This knowledge is represented as a set of allowable transitions (called "inter-word constraints") and a method is given for incorporating this knowledge representation in a system for reading visual images of text. This system first analyses the shape of a word image to suggest a group or *neighborhood* of words in a dictionary (list of words) that contains an input word. The inter-word constraints are then used to reduce the size of the neighborhood and the smaller neighborhood is used to direct further detailed analysis of the input. This process results in a match of the input image to one of the words in the neighborhood. Results are reported in this paper on the performance and cost of two representations for inter-word constraints. The potential of these knowledge sources to reduce the neighborhood size is explored in a series of statistical experiments. It is shown that the average size of a neighborhood encountered when a text of 150,000 words is "read" on a word by word basis is reduced from 16 to about 2. It is also shown empirically that the memory needed for both knowledge representations grows linearly with dictionary size and the total additional memory requirement is about twice that needed for the original dictionary.

1. INTRODUCTION

The development of computer reading algorithms that possess the same versatility as a human reader is a long standing goal of artificial intelligence research. One approach that has been followed in the pursuit of this goal is to determine how humans read and to convert useful portions of this knowledge into algorithms. This method was followed by Shillman[10] in his work on isolated character recognition. He determined the parameters of the human character recognition process and developed algorithms that used these parameters.

The complete, fluent human reading process involves much more than just isolated character recognition. A human reader routinely employs knowledge about the syntactic, semantic, and featural dependencies between words [1, 8]. Although the use of this type of knowledge source in an algorithm for reading text has been suggested [9], no such technique is known.

The use of featural dependencies between words (referred to as inter-word constraints) in an algorithm for reading text is explored in this paper. This knowledge is represented in two ways: either as a table of allowable transitions between words, or as a table of allowable transitions between the featural description of a group of words (called a *neighborhood*) and another word. Both these representations capture a different form of featural dependency. The word-to-word transitions represent knowledge of the words that can follow a given word under the assumption that the given word can be accurately recognized. The neighborhood-to-word transitions represent knowledge about legal successor words under the assumption that the given predecessor word can be approximately recognized. Word-to-word transitions represent the type of knowledge people might use when reading a text for detailed content (e.g. journal article). Neighborhood-to-word transitions are a less specific representation that corresponds to the way people might use featural dependencies in a relaxed reading situation, such as when reading a newspaper.

2. READING ALGORITHM

An outline of the reading algorithm used to evaluate the potential of inter-word constraints is presented in Figure 1. This algorithm assumes that its input is a sequence of word images w_i, $i = 1,2,...$ that make up an input text. These images are input to a **word shape computation** routine that uses a few gross features of a word to determine a neighborhood of words (N_i) in a dictionary (a list of words that could occur in a text) that share that feature set. This neighborhood is then input to the **inter-word constraints analysis routine** that uses either the identity of the previous word (W_{i-1}) or the neighborhood of the previous word (N_{i-1}) along with an internal table of allowable transitions between pairs of words or between neighborhoods and words to produce a reduced neighborhood N_i'. The **hypothesis verification** routine uses the contents of N_i' to direct further selective analysis of the input image, resulting in a recognition decision. The hypothesis verification process is discussed in [5] and will not be extensively analyzed in this paper. Rather, the focus will be on the usefulness of inter-word constraints in reducing the number of words in a neighborhood, thereby simplifying the process of hypothesis verification.

The word shape computation procedure used here (more fully described in [4]) is designed as a generator of candidate words rather than as a complete method for word recognition. This technique is based on the detection of the following six features in a word image:

0. A significant filled space at the beginning or end of a word (e.g., the filled space to the right of the short vertical part in a 'c');
1. A short vertical part (e.g., the leg of an 'r');
2. A long vertical part extending above the main part of a word (e.g., the ascender portion of a 'b');
3. A long vertical part extending below the main body of a word (e.g., the descender in a 'p';
4. Dots over short vertical parts (occurs in an 'i');
5. Dots over long vertical parts (occurs in a 'j');

(Note that these features are specialized for lower case text, however, the technique can be extended to mixed case and upper case input). These features were chosen because they exist in a large

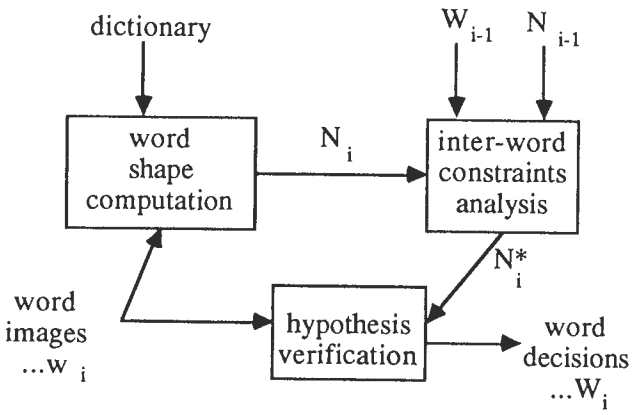


Figure 1. An outline of the word recognition algorithm.

(a) (b)

1. high-vert.
2. short-vert.
3. short-vert.
4. low-vert.
5. short-vert.
6. short-vert.

{"shape", "slope"}

(c) (d)

Figure 2. An example of word shape computation. (a). An input word; (b). The thresholded output of convolution with an 50x5 bar mask; (c). Description of the features detected in (b); (d). The neighborhood was determined from the dictionary of the Brown Corpus [7].

number of lower case fonts and it is easy to design algorithms to detect them. A neighborhood is computed by matching the left-to-right sequence of features that occur in an input word to a similar description of the words in the dictionary. Several classes of errors could be detected at this phase, including incorrect feature extraction and the absence of an input word in the dictionary. An example of the word shape and neighborhood computation is presented in Figure 2. The input word "shape" is shown in Figure 2(a), the result of a convolution operation used to detect vertical parts is shown in Figure 2(b), and a description of each component is shown in Figure 2(c). The neighborhood computed from the sequence of features ("02113110" as discussed above) determined from Figure 2(c) is shown in Figure 2(d). This neighborhood is { "shape", "slope" }.

Two procedures for using inter-word constraints to reduce the size of a neighborhood are explored next. Both of these methods use a table of allowable transitions to reduce the size of the neighborhood of an input word. This representation is somewhat analogous to the thresholded transition probabilities used in the binary n-gram technique for contextual postprocessing [2, 3]. In the first method for reducing the size of a neighborhood, transitions between two words are used. In the second method, transitions between neighborhoods and words are used. The discrete nature of this representation loses probabilistic information, however, it retains significant power, as will be seen.

These representations for inter-word constraints are used to remove words from the neighborhood of a word under the assumption that either the preceding word was perfectly recognized (for word-to-word transitions), or that the neighborhood of the preceding word was correctly computed (for neighborhood-to-word transitions). The use of word-to-word transitions has the advantage of increased accuracy while the use of neighborhood-to-word transitions is insensitive to errors in hypothesis verification that may have been committed on the preceding word. Figure 3. shows a portion of the data structure needed for both these techniques. If word-to-word transitions are used, a word dictionary is augmented by pointers from each word to all the words that can follow it. If neighborhood-to-word transitions are used, a neighborhood dictionary is provided that includes neighborhood-to-word pointers to entries in the word dictionary. For example, if word-to-word transitions were being used when the word "shape" was being recognized in the phrase "... a visual shape ...", and only "shape" can follow "visual" in the corpus, the neighborhood of "shape", (which is {"shape", "slope"}) could be reduced to just "shape" (a perfect recognition). If neighborhood-to-word transitions were employed and both "shape" and "slope" could follow "041112" (the shape number of "visual"), the neighborhood of "shape" would remain {"shape", "slope"}.

3. STATISTICAL EFFECTS OF CONSTRAINTS

The projected performance of both forms of inter-word constraint can be determined by their effect on the number of words that are expected to occur in the neighborhood of a word image. This is measured from a large text by two statistics, the percentage of text that is uniquely specified by shape (PER_UNIQ), i.e., the percentage of a given text that has a neighborhood containing a single word, and the average neighborhood size per text word (ANS_t). Given that $ns(tw_i)$ returns the number of words in the

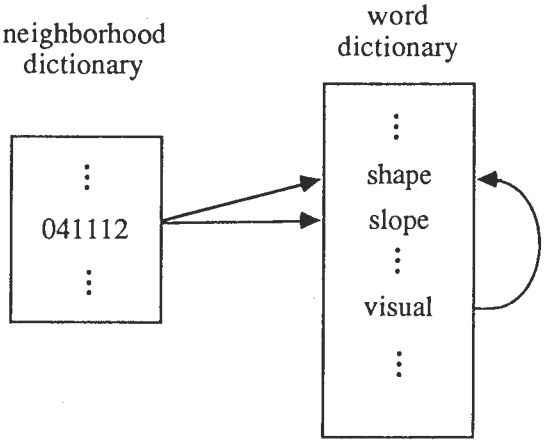


Figure 3. Example of data structures for two forms of inter-word constraint. If word-to-word constraints are used, only the word dictionary and the word-to-word pointers are needed. If neighborhood-to-word constraints are used, the neighborhhod dictionary, the neighborhood-to-word pointers, and the word dictionary are needed.

genre	N_d	N_t	% uniq. isolated	% uniq. nb-wrd	%uniq. wrd-wrd	ANS_t isolate	ANS_t nb-wrd	ANS_t wrd-wrd
A	11,743	88,051	19	71	78	14.1	1.8	1.6
B	8608	54,662	22	74	80	11.4	1.6	1.5
C	7724	35,466	24	82	86	9.7	1.4	1.3
D	5714	34,495	27	79	83	7.0	1.4	1.3
E	9803	72,529	21	73	80	12.8	1.7	1.5
F	12,622	97,658	19	70	78	14.4	1.9	1.7
G	16,126	152,662	15	68	76	16.0	2.1	1.8
H	6628	61,702	24	79	83	7.6	1.5	1.4
J	14,062	160,877	21	72	78	12.8	1.9	1.7
K	8490	58,650	18	70	77	11.9	1.8	1.5
L	6285	48,462	19	72	80	9.7	1.6	1.4
M	2996	12,127	32	85	89	5.7	1.2	1.2
N	8025	58,790	18	71	79	12.0	1.8	1.5
P	7664	59,014	19	71	79	11.9	1.8	1.5
R	4362	18,447	30	84	89	7.1	1.3	1.2
corpus	43,264	1,013,549	9	33	39	38.4	4.4	3.9

Table 1. Summary of the measures of potential usefulness of language context for lower case printed text. The results are broken down in terms of the genres as well as the entire text of the Brown Corpus. Each genre was processed using its dictionary. The allowable transitions were also determined from the genre itself. The percentages of the text uniquely specified by shape and the values of ANS_t are shown when words are considered in isolation, when neighborhood-to-word (nb-wrd) context is used, and when word-to-word (wrd-wrd) context is used. Note that N_d and N_t are the number of words in the dictionary and the text of the indicated genre, respectively.

neighborhood of the i^{th} word in a text,

$$ANS_t = \frac{1}{N_t} \sum_{i=1}^{N_t} ns(tw_i)$$

where N_t is the number of words in the given text. The average neighborhood size per text word thus measures the average number of words that would be presented to the hypothesis verification routine by the word shape computation procedure when it read the given text.

The potential influence of the inter-word constraints on the reading algorithm can be measured by their effect on PER_UNIQ and ANS_t. Since inter-word constraints are supposed to produce a smaller neighborhood, PER_UNIQ should be increased and ANS_t should be decreased as compared to when no inter-word constraints are used. Table 1 shows the results of a study conducted on a collection of over 1,000,000 words of text, known as the Brown Corpus[7], which is representative of contemporary American English. The corpus itself is divided into fifteen individual subject categories or genres. The PER_UNIQ and ANS_t figures were computed over each genre as well as the entire corpus under three constraints. The constraints were: no inter-word constraints, neighborhood-to-word transitions, and word-to-word transitions.

This study shows that there is a large expected improvement in performance when either form of inter-word constraint is employed. In some cases, up to 89% of text is uniquely specified by shape when word-to-word transitions are used and only 30% of the same text is uniquely specified when no inter-word constraints are used. Furthermore, the average neighborhood size per text word is reduced from 38.4 to 4.4 in the entire corpus and from 14.1 to 1.8 in an individual subcorpus when neighborhood-to-word transitions are used and from 38.4 to 3.9 in the entire corpus and from 14.1 to 1.6 in a subcorpus when word-to-word transitions are employed. The fact that these figures drop so dramatically indicates that a large portion of the texts could be recognized by only recognizing the six features discussed in section one and using either form of inter-word constraint. Furthermore, the lack of a significant difference between the two measures indicates that the more specific word-to-word context adds little to the statistical projections. This is satisfying from the reliability point of view since the gains provided by inter-word constraints are insensitive to the results of hypothesis verification. Therefore, it is theoretically possible to recognize a large portion of the texts used in this experiment with only six features and either of the inter-word constraints.

4. EXPERIMENTAL RESULTS

Two experiments were conducted to demonstrate the feasibility of this methodology. The first experiment shows that the shape of a word can be accurately computed under a variety of input conditions that include several different fonts and words with characters touching and overlapping. The second experiment explores the storage required for the transition tables for both representations of inter-word constraints. This is to answer the obvious questions about the practicality of these methods.

The first set of experiments tested the ability to reliably extract the features presented in section one of this paper. If an acceptable degree of reliability can be achieved, it is then reasonable to speculate on the applicability of this methodology to a more general purpose reading situation. The objective of these experiments was not to develop an algorithm for recognizing text in the limited number of fonts currently available, but rather to

show that with a few simple features, that reflect font-independent characteristics, the neighborhoods of words printed in a variety of fonts can be reliably determined. The image database contained five complete 24 point font samples in five display faces (Americana, Baskerville Bold, Bembo, Bodoni Bold, and Bodoni Book) that were digitized on a laser scanner at a resolution of 500 binary pixels per inch. This data was manually segmented into characters and stored in individual files. Words were then generated by appending the appropriate character images.

Two methods were used for feature detection. Dots were determined by locating connected components in the top third of the image. Significant vertical parts were detected by convolution with a vertical bar mask. A response in the thresholded output was considered to correspond to the vertical part of a character only if it was large enough and it was significantly rectangular. Rectangularity was tested by determining if the size of the thresholded response area took up 60% or more of its bounding rectangle. If it did, it was allowed, otherwise it was considered to be a spurious response caused by a slanted part in the image. The dimensions of the mask and the threshold were optimized by manual inspection of the results of the masking operation on a small number of images in each font. These values remained constant for the rest of the experiments. An improved feature testing procedure that was run on twenty fonts with widely varying stroke widths is discussed in [6].

Several experimental runs were made to test the performance of this algorithm. The top 100 most frequent words from the Brown Corpus were used to generate word images. While this is a small sample from the entire corpus, these few words represent 483,355 words or 48% of the entire text. Test images were generated from each word in three ways. The first method merely appended the individual character images from a given font. This is designed to test the general purpose performance of the recognition algorithm on good quality input. The second method for generating word images appended the character images and moved them horizontally until the black portions of their images touched. The third method overlapped adjacent characters by two pixels. The second and third techniques are designed to test performance in a situation that is easy for humans to compensate for but is difficult for a recognition algorithm that requires topologically distinct characters.

The results of these tests are presented in Table 2. Under the first input condition, a 100% correct recognition rate was achieved in all cases except the Bodoni Bookface where a recognition rate of 99% was obtained. However, this did not result in an error since the corresponding shape number did not correspond to the shape number of any other word. When the second condition was tested, worst case performance was only 95% and best case performance was 97%. When overlapping characters were used (the third input condition), worst case performance dropped to 85%, but the best performance was still 97%.

Font	%correct not touch.	%correct touching	%correct overlapping
Amer.	100	96	88
Bas.Bold	100	97	90
Bembo	100	96	96
Bod.Bold	100	97	97
Bod.Book	99	95	85

Table 2. Results of recognition experiment for the top 100 most frequent words in the Brown Corpus generated in five different fonts. The performance under three conditions (all characters topologically distinct, all characters touching, and all characters overlapping by two pixels) is shown.

These results indicate that without much tuning, an algorithm could be developed that accurately recognized the shape number of a word. An inspection of the reasons for the incorrect classifications indicates that with further adjustments of the parameters, correct recognition performance could be increased.

The second set of experiments was designed to explore the storage overhead imposed by the inter-word constraints used here. The memory needed to store the dictionaries as well as the two types of transition tables was determined. The results of this experiment are displayed in Table 3. A full text representation of the dictionary was assumed where every entry is represented by a single byte for each character with an extra byte used to mark the end of a word. The word-to-word transition table was represented by the attachment of a linked list to every dictionary entry, where each element of the list contained a two-byte pointer to another dictionary entry as well as a one byte pointer to the next element in the list. A similar setup was used for the shape-to-word transition table, except that a separate dictionary was maintained for shape numbers. More efficient representations could obviously be designed for both data structures, however, these are suitable for comparison purposes.

The results shown in Table 3 illustrate several interesting points. First, it seems as if the number of links needed for the transition tables are a linear function of dictionary size. This is shown by the two columns headed "links per dictionary word" where the number of links divided by the number of dictionary words is displayed. These values do not fluctuate very much as dictionary size is increased, thus showing that the storage needed for the transition tables at least does not increase as a function of the square of the number of dictionary words, as is theoretically possible.

A further observation about storage requirements is that the number of bytes of memory for the linked list representations in all cases is less than twice that needed for the dictionary itself. This leads to the conclusion that the performance improvement provided by either representation of inter-word constraints can be achieved at reasonable cost.

5. CONCLUSIONS

Two methods of incorporating knowledge about transitions between words (referred to as inter-word constraints) in an algorithm for reading text were explored. The reading algorithm includes a step in which the shape of a word is computed and a group of visually similar words are retrieved from a dictionary. This group of words is then used to control further processing that determines which word matches the input image. Inter-word constraints are used to reduce the number of entries in the initial hypothesis set thus improving the performance of the matching process.

The use of two types of inter-word constraints were explored in this paper. Both of these techniques use tables of allowable transitions as their knowledge source. The first one uses knowledge of shape-to-word transitions and the second one uses word-to-word transitions. The effect of both methods on reducing the number of words that initially match an input word was explored in a series of statistical experiments on a large corpus of text. In all cases the average number of matches was reduced by approximately an order of magnitude to the range of 1.2 to 4.4.

A simulation of the word shape computation illustrated its reliability by showing that 85% to 100% correct performance could be achieved. The storage costs of both representations were also explored and it was found that the size of the transition tables only increases as an approximately linear function of the number of dictionary entries. The total number of bytes needed for the transition tables was shown to usually be about twice the number of bytes needed for the original list of dictionary words.

genre	N_d	$bytes_d$	n-w links	n-w links/ d.wrd.	bytes for n-w links	w-w links	w-w links/ d.wrd.	bytes for w-w links
M	2996	26,071	9280	3.1	27,840	9674	3.2	29,022
R	4362	38,417	12,953	3.0	38,859	13,577	3.1	40,731
D	5714	53,825	23,049	3.9	69,147	23,445	4.1	70,335
L	6285	54,518	28,335	4.5	85,005	30,836	4.9	92,508
H	6628	64,089	33,996	5.1	101,988	36,258	5.5	108,774
P	7664	67,258	34,169	4.5	102,507	37,454	4.9	112,362
C	7724	70,777	25,512	3.3	76,536	27,101	3.5	81,303
N	8025	69,675	34,762	4.3	104,286	38,155	4.8	114,465
K	8490	75,163	34,956	4.1	104,868	38,279	4.5	114,837
B	8608	79,779	35,062	4.1	105,186	37,781	4.4	113,343
E	9803	90,361	45,056	4.6	135,168	49,025	5.0	147,075
A	11,743	107,309	54,023	4.6	162,069	58,733	5.0	176,199
F	12,622	117,945	54,848	4.3	164,544	65,535	5.2	196,605
J	14,062	139,230	82,285	5.9	246,855	90,124	6.4	270,372
G	16,162	155,149	89,199	5.5	267,597	91,144	5.6	273,432

Table 3. The storage cost for the dictionaries and the transition tables. A full text representation for the dictionaries is assumed, and three bytes are needed for each link in the transition tables. $bytes_d$ refers to the number of bytes needed to represent the dictionary. "n-w links" and "w-w links" refer to the number of neighborhood-to-word and word-to-word transitions, respectively.

Therefore, this paper has shown that a relatively simple representation for inter-word constraints can significantly improve the performance of a text recognition algorithm at a relatively modest cost in additional storage. Future work in this area should include additional studies of the storage costs and the development of efficient representations for the transition tables.

Acknowledgements

The author acknowledges the advice of Sargur N. Srihari and the support of the United States Postal Service under the BOA program.

References

1. C. A. Becker, "What do we really know about semantic context effects in reading?", in *Reading Research: Advances in Theory and Practice*, vol. 5 , D. Besner, T. G. Waller and G. E. MacKinnon (editor), Academic Press, New York, 1985, 125-166.
2. A. R. Hanson, E. Riseman and E. G. Fisher, "Context in word recognition", *Pattern Recognition 8* (1976), 35-45.
3. J. J. Hull and S. N. Srihari, "Experiments in text recognition with binary n-gram and viterbi algorithms", *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-4*, 5 (September, 1982), 520-530.
4. J. J. Hull, "Word shape analysis in a knowledge-based system for reading text". *The Second IEEE Conference on Artificial Intelligence Applications*, Miami Beach, Florida, December 11-13, 1985, 114-119.
5. J. J. Hull and S. N. Srihari, "A computational approach to visual word recognition: hypothesis generation and testing", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Miami Beach, Florida, June, 1986.
6. J. J. Hull, *A computational theory of word recognition*, SUNY at Buffalo, Department of Computer Science, 1986. Ph.D. dissertation.
7. H. Kucera and W. N. Francis, *Computational analysis of present-day American English*, Brown University Press, Providence, Rhode Island, 1967.
8. K. Rayner, M. Carlson and L. Frazier, "The interaction of syntax and semantics during sentence processing: Eye movements in the analysis of semantically biased sentences", *Journal of Verbal Learning and Verbal Behavior 22* (1983), 358-374.
9. J. Schurmann, "Reading machines", *Proceedings of the 6th International Conference on Pattern Recognition*, Munich, West Germany, October, 1982, 1031-1044.
10. R. J. Shillman, *Character recognition based on phenomological attributes: theory and methods*, Massachusetts Institute of Technology, August, 1974. Ph.D. Dissertation.

Sensitivity to Corners in Flow Patterns

Norah K. Link Steven W. Zucker

Computer Vision and Robotics Laboratory
Department of Electrical Engineering
McGill University
Montreal, Québec Canada

Abstract. Flow patterns are two-dimensional orientation structures that arise from the projection of certain kinds of surface coverings onto an image. Detection of changes in the orientation structure is an important task since these changes often correspond to significant events such as occluding edges or surface creases. The reconstruction of the flow is not a simple problem however since reliable orientation cues are relatively sparse and are intermingled with conflicting cues. The *assumption of local parallelism* facilitates support between neighbouring cues, thus allowing reliable cues to be reinforced. However the application of this assumption leads to an averaging of orientation cues over large neighbourhoods and therefore results in a loss of sensitivity to changes in the orientation structure. We introduce a new flow parameter – the *path-length of orientation cues*. When path-length increases, more information becomes available locally about both the flow orientation *and* changes in the orientation. This information permits a relaxation of the assumption of local parallelism, effectively reducing the neighbourhood over which averaging occurs when the orientation cues are long. We present psychophysical data to support our argument that the human visual system also takes this parameter into account.

1. Introduction

Vision involves the inference of three-dimensional structures from two-dimensional images. The detection and description of image structures which stand in correspondence with real-world physical structures can aid us in this task ([7], [11], [14], [15]). *Orientation structures* are particularly useful in this regard, since they project onto the image in well-defined ways. There are two basic types of orientation structures: one-dimensional contours which arise, for example, from surface occlusions or creases; and two-dimensional flow patterns which arise from surface coverings, such as fur. In this paper, we shall be concerned with the second type.

Consider a cube covered with fur. Locally, the individual hairs all lie in the same direction, and the overall impression is of a flow, or a two-dimensional orientation structure. While the flow orientation may or may not change within a single face, one would *expect* it to change abruptly along the edge between two faces. Therefore, it is important to be able to detect orientation changes in flow patterns, and in particular to distinguish between smooth and abrupt changes in the flow. The purpose of this paper is to show that human sensitivity to these changes constrains the mechanisms by which people could possibly perceive flow.

Consider the example again, but this time imagine that there are two cubes, one covered with short fur and the other with long; compare the two parts of Figure 1. The shorter pen strokes in part (a) give the impression of a flow in which the small details may change randomly – the face of the cube seems "bumpy". At the same time, the edge between the faces appears to be more rounded, and its exact location more ambiguous, than in the figure produced in part (b) with longer pen strokes. The length of the orientation cues in a flow pattern – which we call *path-length* – thus seems to be a parameter of the orientation information that can be estimated from the image. In this paper, we shall present a model of orientation selection for flow reconstruction which shows why this parameter is significant. In particular, we will show that when the cues are long enough to permit local estimation of curvature, the reconstruction of the flow and the detection of orientation changes, including discontinuities, is more accurate. These conclusions are supported by psyhophysical results.

In the following paragraphs, we briefly outline our model of flow reconstruction. This gives rise to predictions about how sensitivity to orientation discontinuities in flow patterns might vary as a function of path-length. We then present the results of a psychophysical experiment which confirm these predictions.

2. Theoretical Discussion: The Reconstruction of Curves and Flows from Orientation Cues

Informally, a flow pattern is defined as a dense covering of a surface with a family of curves that are locally parallel almost everywhere.[1] Intuitively, one might think of this as arising from a limiting process: consider a surface covered by pin-stripes. Now, imagine adding more and more pin-stripes to the intermediate spaces while at the same time shrinking the width of each pin-stripe. A mathematical idealization of a flow pattern is achieved when the stripes are infinitesimally thin and densely packed. Of course, we shall only be concerned with finite realizations of such processes, but it is instructive to consider the idealization to derive necessary constraints on processes that recover flows. We begin with a preliminary discussion of how orientation structures can be inferred for one-dimensional curves, and then extend this to two-dimensional flows.

2.1 Interpolating Curves Between Dots

The first constraints arise by considering a single curve

[1] For a more precise mathematical presentation, see [15]

Note that an image does not actually contain any curves – which are functions – but just their traces, or the sets of points through which the curves pass. In fact, images that have been sampled by a discrete grid (such as the retinal array) do not contain even the entire trace, but just a subset of the points through which the curve passes. How, then, can a curve be recovered just from sample points? Additional constraints are clearly required, such as how much the orientation of the curve can change between points. These constraints can be described as follows.

Curves can in principle be recovered from their sampled traces by a process called interpolation ([2]). Interpolation theory states that if n positions are represented, then the underlying curve can be approximated by a polynomial of degree $n-1$, and all derivatives of the curve of order n or greater must be assumed to be zero. In practice, however, our system for interpolating curves will have a finite resolution, and will only accomodate, say m derivatives. Then even if the numer of sample points n is larger than m, all derivatives of order higher than m must be assumed to be zero. *Discontinuities* must be asserted at points where this assumption is violated, in order to cause the interpolated curve to pass through the sample points. Therefore, using higher-order estimation procedures results in more accurate placement of discontinuities.

2.2 A Model of Orientation Selection

A scheme for actually computing orientation information is necessary to take the argument further, and we shall use the one described in [15]. This scheme consists of two steps:

i) convolution of the dot pattern against operators that exhibit orientation selectivity ([6]); and

ii) cooperative interpretation of these convolution values to obtain the representation of orientation at each point.

Formally, orientation information is defined to be a field of tangents. Step (i) measures how well a discrete set of possible tangents matches the given points, while step (ii) implements the minimization process that uniformly aligns these original measurements.[2] It is noteworthy that both first- and second-order information – or orientation and curvature – are necessary in schemes such as this one, since it is the estimation of curvature that allows the local tangent estimates to be smoothly aligned. For a formal discussion of how this can be accomplished, see [12]. It also constrains the flexibility of the orientation estimates and therefore assures higher accuracy, especially in the placement of discontinuities. Higher-order information, while potentially useful, is not absolutely necessary. Our psychophysical data suggest, moreover, that it is not used.

[2] The tangent field can be described as the arrangement of abstract unit-length segments that best fits the given points. The 'possible tangents' are represented as unit segments of different orientations at each point.

2.3 The Assumption of Local Parallelism in Flow Reconstruction

The extension from single contours to flows reveals an essential difference between them. While in principle the family of contours is locally parallel almost everywhere, there is no way in practice that such dense patterns can be constructed with finite-sized curves. To keep the orientation cues from overlapping each other, only short segments of each contour are represented. In order to cover the surface (and avoid a pin-stripe effect), the next piece of contour is displayed not for the same contour but for another one displaced slightly in a direction perpendicular to the contour's orientation. The length of each segment, of course, is the path-length parameter. Thus two-dimensional flow patterns, or arrangements of contours, must be discretely sampled not only in the direction of the flow, but also in the perpendicular direction.

The interpolation problem along the flow can be solved similarly to that for single curves, but the interpolation problem perpendicular to the flow must involve different constraints. The most prominent one is local parallelism ([4], [5], [13]), which states that tangent estimates should be averaged perpendicularly. This can be readily incorporated into the minimization procedure described above ([15]).

However, in any region where the flow changes direction – whether smoothly or discontinuously – the assumption of local parallelism is by definition violated. Averaging of orientation cues in these regions must result in a loss of sensitivity to orientation discontinuities; see Figure 2. It is desirable, therefore, to relax the assumption of local parallelism and reduce the neighbourhood size over which orientation cues are averaged when local estimates can be obtained to indicate how the flow direction is changing. The following paragraphs describe how our model accomplishes this.

2.4 Path-Length as a Parameter of Orientation Information

If the model of orientation selection described above is applied to each orientation cue, then the information contained in the path-length parameter becomes apparent. When the cues are sufficiently long with respect to the size of the convolution operators of step (i), curvature (or higher-order) estimates of individual cues can be obtained. These higher-order estimates would then be subject to the co-operative interpretation – or averaging over some neighbourhood – of step (ii). If this interpretation results in non-zero flow curvature estimates, then the assumption of local parallelism can be relaxed and orientation averging reduced. This should result in increased sensitivity to orientation discontinuities with longer path lengths. The increase in sensitivity should be particularly pronounced between very short path lengths, when only orientation information can be derived from each cue, and slightly longer path lengths, when local curvature information can also be estimated.

The psychophysical experiments presented in the following section were designed to investigate the loss of curvature change information as a function of path-length when the assumption of local parallelism is imposed. We are interested

particularly in the perception of *corners*, or aligned discontinuities that are interpretable as surface boundaries (occluding contours) or surface creases (such as a protruding edge).

2.5 Random Dot Moiré Patterns Mimic Natural Flows

The flow patterns that we shall use to study path-length are called *Glass patterns* or *random dot Moiré patterns* ([4]). They are illustrated in Figure 3 and are constructed as follows:

OVERLAY 1:

Construct a field of randomly distributed dots.

OVERLAY n:

i) Make a copy of overlay $n - 1$.

ii) Move each dot in the copy according to a chosen flow transformation, for example a rotation or a translation.

iii) Superimpose this overlay on the other overlays.

Only two overlays ($n = 2$) are required to produce the impression of flow. We define the number of overlays used, n, as the *path-length metric* or *path metric* of each stroke. The path metric times the dot separation gives the actual path-length of a stroke.[3] Through interpolation theory, this metric also provides a direct and meaningful measure of what orientation information is available locally and therefore allows us to determine how changes in orientation could ideally be processed at different path-lengths.

3. Experiment: Corner Sensitivity vs. Path Metric in Random Dot Moiré Patterns

The purpose of this experiment was to determine how the path-length of a flow pattern affects the sensitivity of human observers to changes in curvature of the underlying flow. Specifically, we were looking for the lower bound of orientation changes for which discontinuities or corners in the underlying flow can be detected, as a function of the path metric in random dot Moiré patterns

Subjects were shown several instances of random dot Moiré patterns derived from two types of flow. These consisted of two translational flow regions meeting in the centre either at a discontinuity or with a smooth transitional region of flow: see Figure 4. The experiment was designed to evaluate how well subjects could distinguish between the two types. Several differences in orientation between the two translational flow regions were used. The independent variable in the images was the path metric, or the number of overlays used to produce the patterns. Sample test images from the experiment are shown in Figure 5. Subjects were given three possible responses: a single abrupt change in orientation, a smooth change in orientation, or ambiguous. Other factors which could affect sensitivity to discontinuities, such as dot density, dot size, and dot spacing, were held constant at a value well within the range that permits these patterns to be clearly visible.[4] Note also that by keeping the dot spacing constant, each increase in path metric corresponded to a similar increase in the path-length – each cue did not become "denser" or more "solid". For experimental details, see [9].

The results for discontinuous flow fields are tabulated for each path metric in Figure 6. We show the average percentage of discontinuities that were correctly perceived, as a function of the orientation change. A Chi-square test indicates that the magnitude of this change significantly affected the responses of all subjects at all path lengths ($p = 0.999$). The 50% threshold sensitivity to discontinuities is plotted as a function of path metric in Figure 7. Note that the most significant change in threshold occurs between path metrics of 2 and 3 dots, where it decreases from 27.0° to 11.4°. (The threshold sensitivity for dotted one-dimensional contours with the same dot size and dot spacing is also 11.4° [10].)

Consider the flow patterns with orientation changes between 13.69° and 18.18° that were presented with path matrics 2, 3, and 5. A Chi-square test indicates that the path metric for these images had a significant effect on the responses for all subjects ($p > 0.999$). In this same region, the actual change in orientation had no effect on the response ($p < 0.003$), confirming that these angles actually lie between the 50% thresholds for path metric 2 and the other path metrics; see Figure 7.

4. Conclusions

In conclusion, then, we assert that the anticipation and detection of changes in orientation structure within flow patterns is essential to accurate recovery of the flow. On the surface, our theory is similar to others that have been proposed for flow reconstruction ([4], [5], [13]). These theories also involve the two stages of extraction of orientation cues followed by an interpretation of these cues, using the assumption of local parallelism. But it is in the interpretation stage that our theory differs.

Rather than rigidly applying assumption of local parallelism, our method seeks to impose a structure *in the direction of the flow* which allows the assumption of local parallelism to be relaxed in regions of (systematic) orientation change. The degree to which we can relax the assumption depends directly on the information we have about the structure of the flow. We can take advantage of curvature information available at longer path-lengths only because we have a mechanism for representing the local structure of the flow in *two dimensions* – in the direction perpendicular to the flow (the notion of local parallelism) *and* in the same direction as the flow. Our experimental results demonstrate that the human visual system also takes both dimensions of the flow structure into account when it reconstructs the flow, and that the path-length of orientation cues provides an important information parameter

[3] The issue of the functional equivalence of these patterns to "continuous" flow patterns is discussed in more detail in [8] and [9]

[4] Preliminary experiments showed that increasing the path metric had a more profound effect on curvature sensitivity than increasing the dot density and that they also fall within the size/separation constraint for the related problem of one-dimensional contour inference ([15])

This research was supported in part by NSERC grant A4470. We wish to thank Pierre Lamoureux for programming assistance; and Yvan Leclerc, Pierre Parent, Frank Ferrie, Kamal Gupta, Nevine Nassif, and Paul Freedman for many helpful discussions.

Dr. Zucker is a Senior Fellow of the Canadian Institute for Advanced Research.

References

[1] Attneave, F. 1957. Physical determinants of the judged complexity of shapes. *J. Exper. Psych.* 53:221-227.

[2] Davis, P.J. 1965. *Interpolation and Approximation*. New York: Blaisdell Publishing Co.

[3] Freund, J.E. 1971. *Mathematical Statistics*. 2nd ed. New Jersey: Prentice-Hall.

[4] Glass, L. 1969. Moiré effect from random dots. *Nature* 223:578-380.

[5] Glass, L. 1982. Physiological mechanisms for the perception of random dot Moiré patterns. In *Chaos and Order in Nature: Proc. of the Int. Symp. on Synergetics*, at Schloss Elmau, Bavaria, April 27 – May 2, 1981. ed. H. Haken. (*Springer Series in Synergetics*, Vol. 11) Berlin: Springer-Verlag.

[6] Hubel.D., and Wiesel, T. 1977. Functional architecture of macaque monkey visual cortex. *Proc. Roy. Soc. (London), B* 198:1-59.

[7] Koffka, K. 1935. *Principles of Gestalt Psychology*. New York: Harcourt, Brace & World.

[8] Link, N.K. 1985. *Curvature Cues and Discontinuity Detection in Early Orientation Selection*. Masters Thesis, Department of Electrical Engineering. Montréal: McGill University.

[9] Link, N.K., and Zucker, S.W. 1985. Sensitivity to Corners in Flow Patterns. *Technical Report* 85-4R. Montréal: Dept. of Elect. Eng., McGill University.

[10] Link, N.K., and Zucker, S.W. 1985. Corner detection in curvilinear dot grouping. *Technical Report* 85-5R. Montréal: Dept. of Elect. Eng., McGill University.

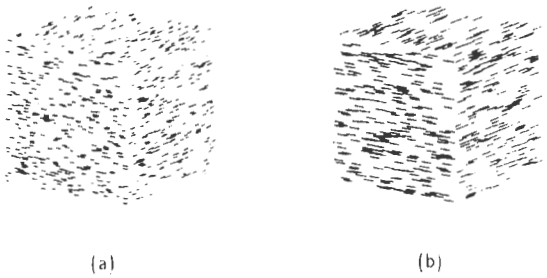
[11] Marr, D. 1982. *Vision*. San Francisco: W.H. Freeman & Co.

[12] Parent, P., and Zucker, S.W. 1985. Curvature consistency and curve detection. *Technical Report* 85-12R. Montréal: Dept. of Elect. Eng., McGill University.

[13] Stevens, K.A. 1978. Computation of locally parallel structure. *Biol. Cyber.* 29:19-28.

[14] Witkin, A.P., and Tenenbaum, J.M. 1984. On the role of structure in vision. In *Human and Machine Vision*. eds. A. Rosenfeld and J. Beck, pp. 481-543. New York: Academic Press.

[15] Zucker, S.W. 1985. Early orientation selection: tangent fields and the dimensionality of their support. *Technical Report* 85-13R. Montréal: Dept. of Elect. Eng., McGill University. *Comp. Vision, Graphics, and Im. Proc.*, in press.



(a) (b)

Figure 1 A cube described by flow patterns. There are more orientation cues in part (a) than in part (b), but they are also shorter. Note that the edge between the front faces is not as sharp in (a) as in (b).

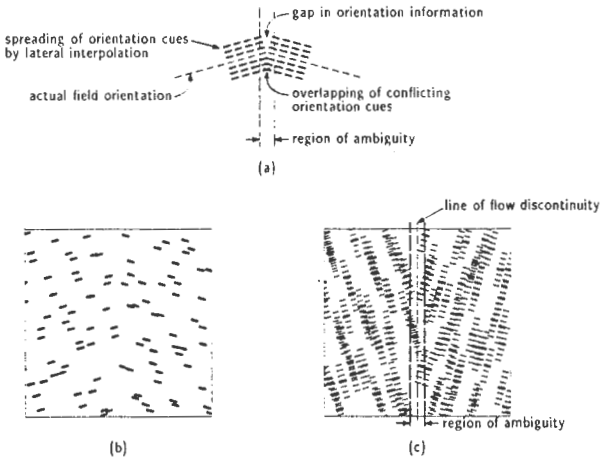
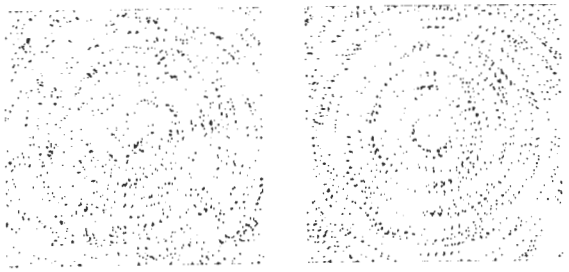
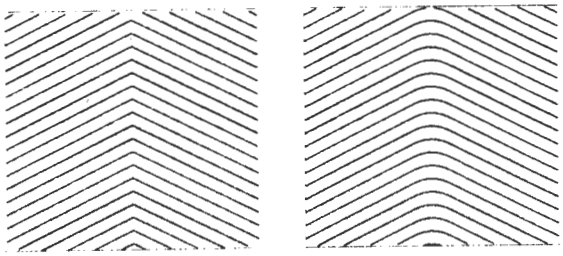


Figure 2 The ambiguity introduced by the assumption of local parallelism at changes in orientation. The lateral propagation of support for the orientation cues results in conflicting orientation support inside the corner, and a gap in orientation support outside the corner which compounds these ambiguities.



(a) (b)

Figure 3 Sample random dot Moire patterns created by rotating each dot in the previous overlay about the centre of the figure. Part (a) has two overlays, part (b) has three overlays. Both parts have the same overall density and the same distance between correlated dots.



(a) (b)

Figure 4 The flow transformations used in the experiment to distinguish between orientation discontinuities and smooth orientation changes

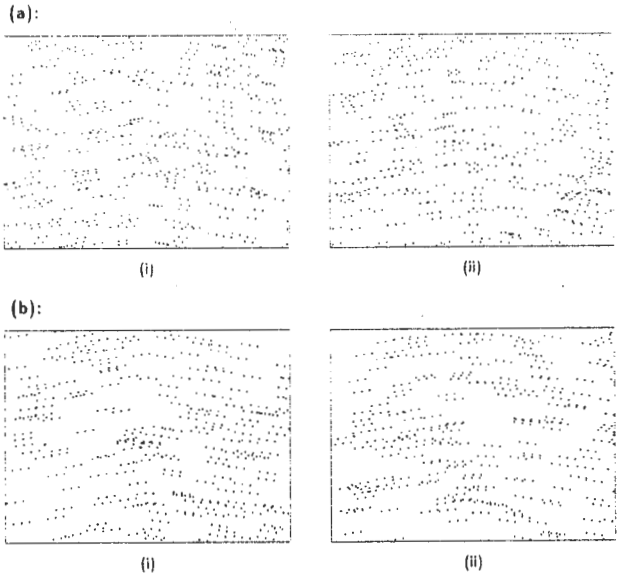


Figure 5 Sample test images for the experiment. a) path metric 2. b) path metric 3. Part (i) of each is a discontinuous transformation with an underlying angle of 13.69° part (ii) is the smooth transformation for the same overall change in orientation

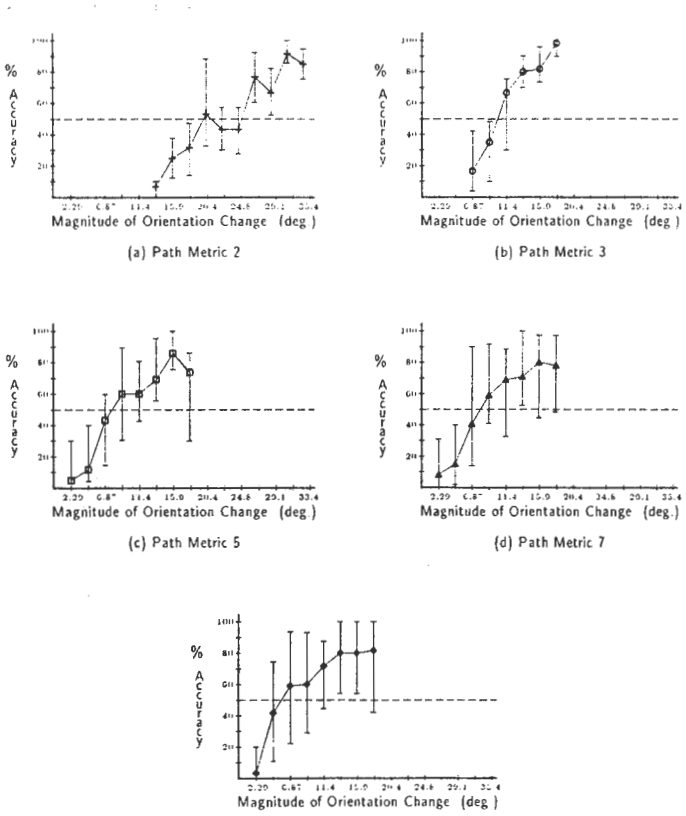


Figure 6 The results by path metric of the experiment Illustrated are the mean percentages across all subjects (and one standard deviation) of flow patterns with discontinuities that were correctly identified, as a function of the magnitude of the orientation change. (Note that all smooth patterns were correctly identified.)

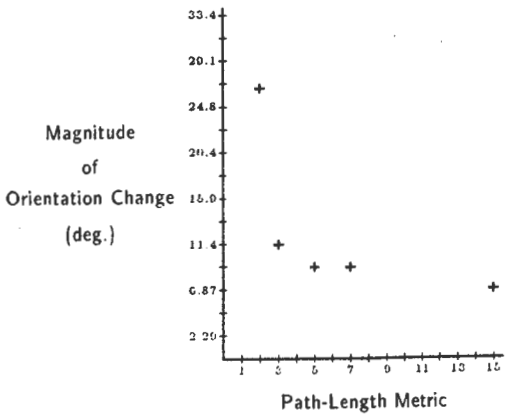


Figure 7 The mean threshold sensitivities (50% accuracy of detection) to discontinuities in flow patterns, as a function of path metric.

Stable Surface Estimation

Peter T. Sander
Steven W. Zucker*

Computer Vision and Robotics Laboratory
Department of Electrical Engineering
McGill University
Montréal, Québec, Canada
* Senior Fellow Canadian Institute for Advanced Research

Abstract

The surface of an object in a 3D image is determined from its trace, the set of points on the surface. We use the local geometric structure in the image, based on a quadric surface model, to estimate the trace points, and use the condition number of the least-squares fit to the quadric to determine the validity of the model. We show how these local estimates can be combined into consistent traces.

Résumé

Pour bénéficier du plein potentiel des nouvelles techniques d'imagerie tri-dimensionnelle (3D) non invasives, il est nécessaire de retrouver la structure des objets dans ces images. La surface d'un tel object est calculée à partir de son support, c'est à dire l'ensemble des points sur la surface. Nous montrons comment déterminer un support consistant. La structure géométrique locale, basée sur un modèle quadratique local, est évaluée à chaque point, et nous proposons des techniques de l'analyse numérique (voire le conditionnement de l'erreur quadratique minimum) pour évaluer la validité du modèle. Cette information locale est ensuite organisée en un support consistant, qui permet d'obtenir l'expression analytique de la surface.

1. Introduction

The increasing presence of three-dimensional non-invasive imaging, e.g., nuclear magnetic imaging or computer tomography, makes the problem of inferring the structure of depicted objects more pressing. Such object descriptions are necessary for both viewing internal organs and for correlating data from multiple scanners.

3D images avoid the ambiguity introduced by projection from a 3D world to a 2D image, however, many of the fundamental problems of image understanding remain (boundary point detection, organization of points into object boundaries object determination and identification) and are complicated by the additional dimension. In this paper, we concentrate on the problem of recovering object surfaces from the 3D data.

A piece of object surface is described as a mapping $\mathbf{s} : U \subset \Re^2 \rightarrow \Re^3$. As a step towards determining this mapping, we first locate its *trace*, i.e., the set of data points through which the surface passes. More specifically, let (x, y, z) denote a location in the 3D image, then

$$trace(\mathbf{s}) = \{(x, y, z) \in \Re^3 \mid \mathbf{s}(u, v) = (x, y, z), (u, v) \in U\}.$$

However, the 2D trace is not given explicitly (as in range-finding techniques [1]) and must first be extracted from the 3D data.

We assume that the objects of interest have piecewise smooth surfaces. In particular, we assume that locally the surface model is a 'constrained' quadric, since this is the lowest order function from which curvature properties can be computed. Our particular goal in this paper is to show how these properties can be computed reliably and stably. The result is a better estimate of the trace points and the local differential geometric structure which can then be assembled into an analytic mapping.

We first obtain estimates of the surface orientation and curvature at putative trace points based on the above quadratic model. But such models are not appropriate everywhere, and we introduce techniques from numerical analysis to indicate where other models should be considered. Finally, we show how these local estimates can be combined into consistent traces.

2. Local information

The first step in our analysis determines which points of the three-dimensional data space can yield further information about the structure of the world by separating points which might belong to some object trace from background and interior points. In ananlogy to detecting likely edge points in 2D images, we are seeking likely surface points. A 3D gradient operator [2] associates a surface normal $\mathbf{N_p}$ with points $\mathbf{p}$. Points producing large operator response are the best candidates for surface points, as are those in 2D images with large 'edge detector' responses.

How can the true trace points be extracted from the set of candidate trace points? If we had the surface mapping, then it would be a simple matter to check them. But, of course, we do not — it is this mapping that we are after. We use local techniques to construct one. $\mathbf{N_p}$ provides sufficient information to approximate the mapping at $\mathbf{p}$. We consider that $\mathbf{N_p}$ defines a local tangent plane and fit a local quadric surface

$$\begin{aligned}\mathbf{s}(u,v) &= (x(u,v), y(u,v), z(u,v)) \\ &= (u, v, au^2 + buv + cv^2)\end{aligned}$$

(u, v in tangent plane coordinates) to a neighbourhood of $\mathbf{p}$. The least-squares fit

$$A\mathbf{x} = \mathbf{b} \tag{1}$$

determines quadric parameters $\mathbf{x} = (a, b, c)^T$ from the coefficient matrix

$$A = \begin{pmatrix} u_1^2 & u_1 v_1 & v_1^2 \\ \vdots & \vdots & \vdots \\ u_n^2 & u_n v_n & v_n^2 \end{pmatrix}$$

and data vector $\mathbf{b} = (z_1 \ldots z_n)^T$ at local neighbourhood points $(u_i, v_i, z_i)_{i=1}^n$ of the image.

The Gaussian and mean curvatures, K and H, of the local surface at $\mathbf{p}$ can be computed from the quadric mapping. $\mathbf{p}$ can be classified by its curvature type as elliptic ($K > 0$), hyperbolic ($K < 0$), parabolic ($K = 0$ but $H \neq 0$), or planar ($K = H = 0$). We shall need these classifications to piece the local quadrics together and, eventually, to segment larger-scale objects into composite parts.

3. Validation of local information

Before using the local results, the local assumption of a quadric model must be validated. A measure of the reliability can be obtained from a closer consideration of the fit of the quadric model to local candidate trace points. Small errors will arise unavoidably from many sources (noise, sampling and quantization, round-off, etc.) and we expect that they should only make small differences to further computations. On the other hand, applying methods developed for our underlying locally smooth model to parts of the image where the model is not applicable will render them quite invalid, e.g., in determining the curvature of a quadric surface fit to a spike.

The approach to validation is to show that the estimates are stable. If the local configuration of data were different due to error, i.e., if we were solving

$$(A + \epsilon A)\hat{\mathbf{x}} = \mathbf{b} + \epsilon \mathbf{b},$$

instead of equation (1), we expect only small changes in the solutions $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$ for small differences δA, δb. Bounds on the relative error are given by

$$\frac{1}{cond(A)} \frac{\mathbf{r}}{\mathbf{b}} \leq \frac{\mathbf{e}}{\mathbf{x}} \leq cond(A) \frac{\mathbf{r}}{\mathbf{b}} \tag{2}$$

in terms of the residual of the fit $\mathbf{r} = A\mathbf{x} - \mathbf{b}$ and the condition number of the coefficient matrix

$$cond(A)^2 = cond(A^T A) = \frac{\max \lambda(A^T A)}{\min \lambda(A^T A)},$$

for eigenvalues λ.

As equation (2) shows, the residual by itself is not a good enough bound on the error since an ill-conditioned A (i.e., an A with large condition number) may have both a small residual and a large error. The condition number is an indicator of the inherent stability of the system to small perturbations. Thus, an ill-conditioned A in the least-squares fit at $\mathbf{p}$ means that one must be cautious in applying the quadric model there, and that the behaviour of the surface at $\mathbf{p}$ should be investigated using other models. To illustrate, Figs. 1 and 2 show local regions to which least-squares fits approximately planar surfaces and determines similar relative residuals. It is the very large condition number for Fig. 2 which indictes that the fit is unstable (positional information indicates a plane viewed edge-on while surface normal information indicates a plane facing the viewer).

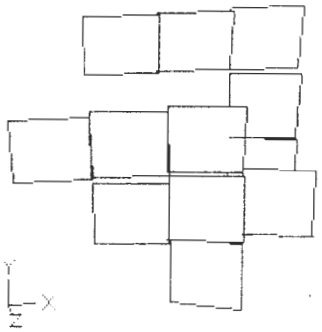


Figure 1 Tangent planes to which least-squares fits an approximately planar region with relative residual 0.99 and condition number 1.69

The quality of the local results affects the success of further processing since the results from inappropriate models have been eliminated

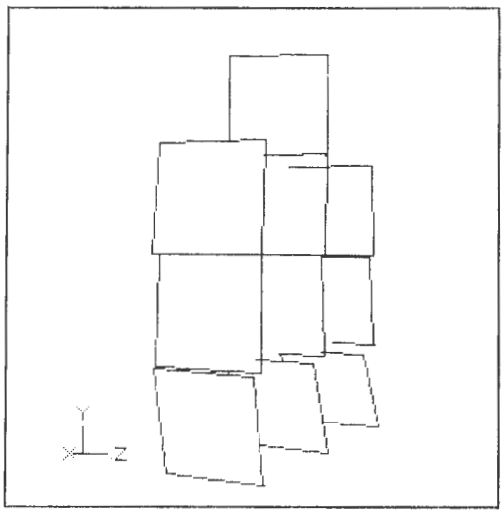


Figure 2 Tangent planes to which least-squares fits an approximately planar region with relative residual 0.98 and condition number 29.3. Point positions indicate a plane viewed edge-on: surface normal indicate a plane facing toward the viewer

4. Grouping

So far we have been estimating only local properties of the surface mapping (orientation and curvature type) at candidate trace points. The next step is to organize this local information so that more global structure emerges. In particular, we demand consistency between candidate trace points in small neighbourhoods, where consistency is defined in terms of our assumed piecewise smooth **s**. Local properties of consistent candidate trace points can be refined and inconsistent points eliminated. This has the effect of both overcoming errors in the data (in the sense of the previous section) and of providing a foundation for partitioning the surfaces into meaningful pieces [3],[4].

Thus, at this stage we move beyond local computations to consider relations between points over neighbourhoods. In particular, two salient features of the surface model provide neighbourhood consistency constraints: (i) smoothness ensures that the surface normal varies continuously almost everywhere, which implies (ii) that regions composed of elliptic points (where the Gaussian curvature is positive) are separated from regions of hyperbolic points (where it is negative) by lines of parabolic points (where it is zero and the mean curvature is nonzero).

We exploit the observation that, for smooth surfaces, Gaussian curvature will vary slowly. In particular, we assign curvature labels

$$l \in \{elliptic, hyperbolic, parabolic, planar\}$$

to candidate trace points **p** to minimize an appropriate functional. Letting n be the number of points in the neighbourhood of **p**, n_l the number of points in the neighbourhood with label l, a suitable functional is $\mathcal{L} = \sum_{\mathbf{p}} L$ where

$$L(l_{\mathbf{p}}, n_l) = \begin{cases} 0 & \text{if } l_{\mathbf{p}} = \hat{l}, \text{ where } n_{\hat{l}} \approx n; \\ 0 & \text{if } l_{\mathbf{p}} = parabolic, \\ & \quad \text{and } n_{elliptic} \approx n_{hyperbolic} \approx \frac{1}{2}n; \\ 1 & \text{otherwise.} \end{cases}$$

The updating process is repeated iteratively until convergence [2]. Figs. 3 and 4 show the consistent labeling surface points of a synthetic image after a small number of iterations.

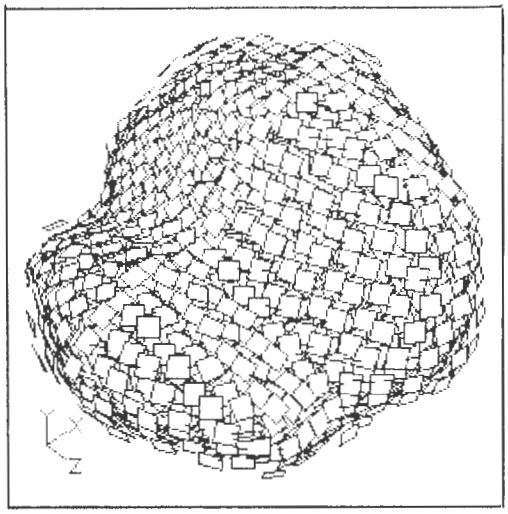


Figure 3 Local tangent planes of a synthetic 3D image.

The result of increasing neighbourhood consistency is twofold. First, mis-labeled points within similarly-labeled neighbourhoods are updated to reflect the overall character of the neighbourhood. Second, the label *parabolic* is assigned to those points which lie between regions of different types. These Gaussian curvature neighbourhoods separated by parabolic lines are exactly the traces that we need. They partition the surface into meaningful pieces and from them the corresponding mappings are readily determined (e.g., by least-squares fitting a quadric or superquadric [5],[6] model). Figs. 5 and 6 show the organization of points into traces with qualitatively-similar curvature labels for nuclear magnetic resonance data.

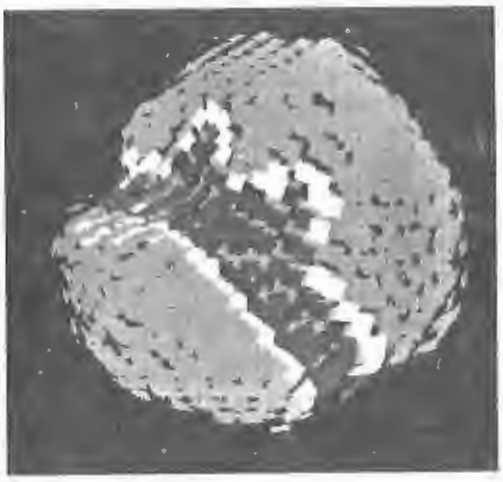


Figure 4 Classification of surface points after four iterations. Dark points are hyperbolic grey are elliptic, white are parabolic.

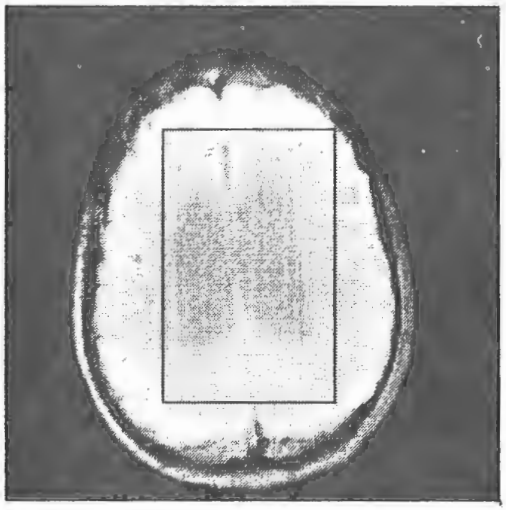


Figure 5 Magnetic resonance imaging section with outlined lateral ventrical

5. Conclusions

We have demonstrated a working system to extract and partition the surface of objects in three-dimensional data. In determining first and second differential properties of possible surface points, the method carefully examines stability information to judge what model of the local world to use. These computed local properties are then used to assemble the points into explicit traces of surface patches from which a meaningful partition into analytic surface mappings is derived

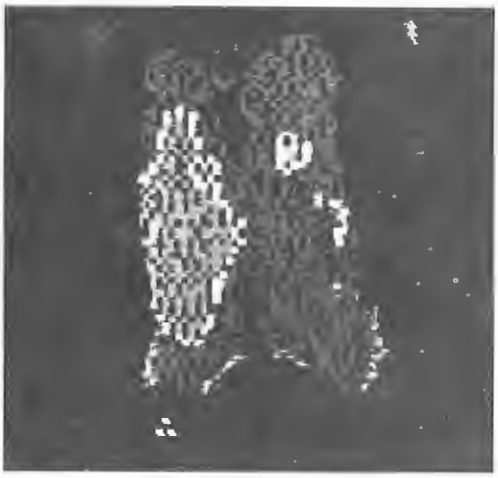


Figure 6 Classification of surface points of lateral ventricals. Dark points are hyperbolic, grey are elliptic, white are parabolic.

6. References

[1] O.D. Faugeras, and M. Hebert, "The Representation, Recognition, and Positioning of 3-D Shapes from Range Data", draft, INRIA, Rocquencourt, 1985.

[2] S.W. Zucker, and R.A. Hummel, "A Three-Dimensional Edge Operator", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-3, No. 3, May 1981, pp.324–331.

[3] D.D. Hoffman, and W.A. Richards, "Parts of Recognition", *Cognition*, Vol. 18, 1984, pp. 65–96.

[4] J.J. Koenderink, and A.J. van Doorn, "The Shape of Smooth Objects and the Way Contours End", *Perception*, Vol. 11, 1982, pp. 129–137.

[5] A. Barr, "Superquadrics and Angle-Preserving Transformations", *IEEE Computer Graphics and Applications*, Vol. 1, pp. 1–20

[6] Pentland, A.P., "Perceptual Organization and the Representation of Natural Form", SRI Technical Note 357, July, 1985.

Measuring Motion in Dynamic Images: A Clustering Approach

Amit Bandopadhay & R. Dutta

Computer Science Department
University of Rochester

ABSTRACT

An algorithm to match interest points across multiple time frames of time varying images is described. Interest points are found by decomposing an image patch at a location into a weighted sum of orthogonal basis functions. Subsequently, a simple quantitative decision rule, based on the above weights, is used to select or reject a candidate location. The match values in a spatial neighbourhood are assumed to be similar to each other. The set of matches is computed by considering all possible associations in neighbourhoods of fixed size. The algorithm proceeds in two stages. In the first phase, clustering in the match vector histogram determines plausible population of the optical flow space. Next local similarity is used to select the correct match at every point.

1. Introduction

Image motion measurement techniques can be classified as follows:

1. *Differential Methods:* The motion is computed from spatio-temporal gradients of the image intensity function, or equivalently, from local measurements of spatio-temporal frequency patterns in the image intensity distributions [10, 12, 14, 18, 21, 22]. Another approach is to aggregate motion measurements along contours [13, 27]
2. *Discrete Methods:* The formulation in this case concerns the measurement of inter (time) frame image-point or feature displacements. This class of algorithms attempt to solve the so called correspondence problem. Many of these algorithms use relaxation to converge to a global matrix of disparity values compatible with local constraints [4, 23]

In general, the above methods are specialized and work reasonably well only in restricted domains. Examples of restrictions imposed are: uniform illumination, smoothly varying reflectance, being able to locate smooth zero crossings (of the Laplacian of the image intensity) contours. One shortcoming of these approaches is that they deal primarily with movement of a single object in the environment or motion of the observer. Some of the above techniques are sensitive to noise. The present approach seeks to remedy these lacunae.

The scheme of computation of the retinal motion has the following steps :

(1) Location of points in the image with significant intensity variation (contrast). The goal is to select locations in the image where there is significant (with respect to its neighbourhood) contrast, yet there is no orientation specificity. This means that two criteria must be satisfied:

(a) The contrast variation at and around the selected location should be high. This is measured by means of a centre surround operator like the Laplacian or DOG operator [17, 19]

(b) A good edge operator should not respond to intensity distribution at the same location, or have multiple weak responses.

The interest operator that was used is described in [3]. This operator decomposes the local intensity distribution around a point in the image into a set of basis functions. The selection of the point then depends on the relative responses of the "edge" and "extremum " subspaces in the basis set. (Similar techniques may be found in [11, 15])

(2) *Measurement of image motion*: It is assumed that the velocity field is smooth, except for a small number of places where motion bounderies occur. Each point selected at a given time instant, attempts to find a match in a point, selected at a later instant, that is in its neighbourhood (see figure 1). To determine the goodness of the match, each of these "velocity units" (or plausible match vectors) evaluate the support it receives from nearby velocity units. This scheme was used with great success for stereo matching by Prazdny ([24]). Finally only those matches that can muster more support than competing matches get selected.

(3) *Clustering in image motion space*: The dimension of a match vector is four. These correspond to two cartesian coordinates (x,y) for starting position of the vector and two other quantities representating its magnitude and orientation, i.e. (ρ,θ) . The uniformity in the field of match vectors is made explicit by clustering. However, to reduce the computational complexity of the clustering process we first define a cluster space by projecting the four dimensional vectors onto a space of lower dimensions. Generally the cluster space dimension is two. It is formed by dropping the position coordinates from the four dimensional vectors. The elements in the cluster space group around similar (the Euclidean distance metric is used) units and support each other. This helps to remove the outliers among them. The units that belong to some cluster are then retained and the rest are deleted. The surviving units then mediate the matching process in the four dimensional space of match vectors.

The discussion so far would seem to imply the requirement of two "snapshots" of a dynamic scene, taken at consecutive time instants. This is not a critical aspect of the method, being only used for ease of explanation. In fact it adapts very easily to a sequence of temporal frames of a changing scene. In this case all we need to do is to introduce a temporal decay rate for the accumulated support for the match units. The algorithm has been tested with synthetic images comprising spheres and planes, which where painted with random dot patterns. This was mainly done so that the computed motion vectors could be compared with the actual values. The experiments with synthetic data show that multiple moving bodies and random noise points of upto 20% can be handled by the algorithm. The algorithm was also tested on a real world scene with good results. The following sections detail the various sections of the algorithm and the experimental results.

2. Computing Interest Points

An interest point is a point in the image (actually a small neighbourhood) which has *properties* that distinguish it from its neighbouring points. The properties in question may be simple, like gray levels, or sophisticated ones indicative of the local topography of the imaged surface. Previous approaches to finding interest points are exemplified in the work of Moravec, Kitchen & Rosenfeld, Nagel, Davis, Sun & Wu, Fang & Huang ([7, 9, 16, 20, 21]). The above approaches utilize operators that are nonlinear and require, in some, computation of higher order spatial derivatives of the image function. Another method is a locationwise *topological classification* of the image function. However, topological analysis is computationally intensive. Furthermore, this method has to deal with the *Hessian* of the image function at every point. Hence, it is also inherently unstable in the presence of noise. A crucial observation ([5]), is that, interesting patterns in the image can be thought to have a sharply peaked *autocorrelation function* . This observation gives rise to a practical interest operator which selects image locations whose autocorrelation decays sharply with increasing eccentricity. This could be implemented with great simplicity if we could design matching templates with the above property. However no constructive algorithm exists to automatically generate this type of template. In any case, sharply diminishing autocorrelation is a desirable property for operator templates, and it is useful to bear this fact in mind.

The location of interest points is the first step in tackling the so called correspondence problem. Hence the operator must satisfy 3 requirements:

1. Selected points must be sparse
2. Contours must be suppressed
3. Interest points should be, largely, stable across frames

The method (described in [3]) is computationally simpler than the prevalent techniques at locating interesting points in images. The image function is decomposed into a weighted sum of basis functions. The central idea being that if the basis is chosen with care, then the distribution of the respective weights indicates the nature of the image function directly. A similar approach can be seen in the literature in other image processing contexts ([11, 15]).

3. Algorithms for retinal motion measurement

3.1. The Matching Algorithm using local support

The *corrrespondence problem* is almost universally regarded as difficult. As mentioned earlier, it arises in the measurement of image *disparities*. The problem is magnified for motion measurement, since the disparity in this case is not constrained, as in the case of stereo, to be parallel to a base line. The overall scheme of thing is simple: select interest points in image frames and then decide which point from one frame matches another point from the other frame. If it is possible to obtain interest points that are sparse then correspondence is not difficult. Here sparseness means that the average disparity value is smaller than the average spatial distance between points in the same image frame. An interesting quantification for the degree of sparseness is due to Kent Stevens [25] and is the number of false matches possible, on average, for a given match neighbourhood size.

The algorithm proceeds from the interest point stage by forming all possible matches subject to a maximum limit on the magnitude of the match vector. This is equivalent to saying that the match neighbourhood size is determined a priori. Each match vector then proceeds to accumulate evidence supporting its existence within a support neighbourhood, which is larger than the match neighbourhood. This scheme is based upon the assumption that the imaged surface depth varies smoothly (a similar scheme is reported to be successful with the stereopsis problem [24]).

Consider optical flow (u,v) generated by a translating object. The constraint equations are

$$u = \frac{U - xW}{Z}$$

$$v = \frac{V - yW}{Z}$$

where (U,V,W) is the translational velocity in three space, Z is the depth of the object corresponding to the retinal location (x,y).

If the depth function $Z(x,y)$ is smooth then, to a first order approximation, the spatial rate of change in the optical flow is proportional to the spatial rate of change in depth. For instance consider the optical flow value at $p = (x_0,y_0)$. Let it be (u_0,v_0), also let the depth at p be Z_0. Then the difference between optical flow at p and a neighbouring point $r = (x,y)$ is :

$$\delta u = u(x,y) - u_0 \simeq \frac{W}{Z_0^2}\Delta x + \delta Z\left[\frac{U - x_0 W}{Z_0^2}\right]$$

where $\Delta x = x - x_0$ and higher order terms involving $\frac{\Delta Z}{Z_0}$ and $\frac{\Delta x}{Z_0}$ are neglected. This leads to:

$$|\delta u| \leq \lambda_1 |\delta Z| + \lambda_2 |\delta x|$$

$$|\delta v| \leq \lambda_3 |\delta Z| + \lambda_2 |\delta y|$$

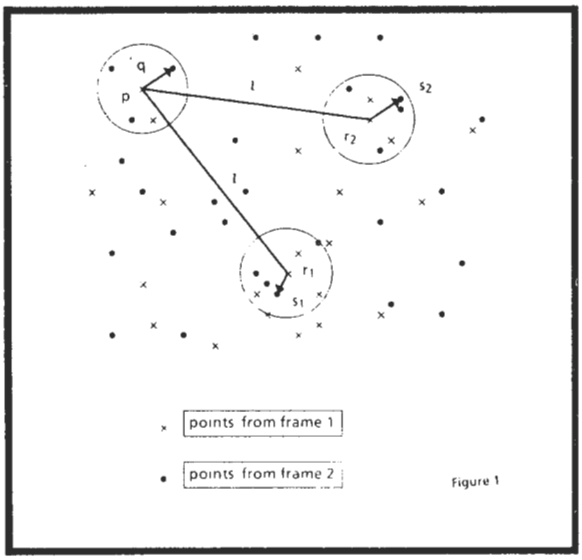


Figure 1

Superimposed points of the second frame on the first frame
The vector $r_2 s_2$ provides more support to the vector pq than the vector $r_1 s_1$.
The difference between the vectors r, s, and pq gives the value of d for calculating support. l is the distance between the origin of the two vectors. The support can be computed by applying the appropriate support function

where λ_1,λ_2 and λ_3 are constants for a local image neighbourhood. Combining the above we have:

$$\frac{dst(u,v)}{dst(x,y)} \leq \lambda \frac{|\delta Z|}{dst(x,y)} + \lambda_2$$

where $\lambda = \lambda_1 + \lambda_3$, $dst(u,v) = |\delta u| + |\delta v|$, and $dst(x,y) = |\delta x| + |\delta y|$. Thus the situation that arises here, is that the smoothness in the depth function relates to the smoothness of the displacement field The support that two candidate vectors (u_1,v_1) and (u_2,v_2) at retinal locations (x_1,y_1) and (x_2,y_2) respectively provide each other is given by the function $S[dst(u,v),dst(x,y)]$. Experimental results indicate that a linear support function is adequate. It should be noted that in Prazdny's algorithm for stereopsis [24], an exponential support function is used. The support function is a quantitative expression for the notion of local smoothness. Prazdny's choice of support function is intuitive, based on psychophysical data. The same justification applies to motion correspondence. As an example consider the following exponential support function, which we used in our experiments:

$$S(d,l) = \frac{1}{l} e^{-(\frac{d}{l})^2}$$

where $l = dst(x,y)$ and $d = dst(u,v)$. As mentioned previously, there seems to be no great advantage in using an exponential support function in preference to a linear one. The advantage of clustering is that, once the clusters have been determined, the parameters of the support function are obtained. Thus once we know that the maximum disparity difference (i.e. the diameter of the cluster), we can calculate the largest disparity gradient that should be allowed. This is the ratio of the cluster diameter (i.e. largest linear distance across the cluster) and the diameter of the support window in image space. Suppose this ratio is Δ and $f(x,y) = \frac{1}{y\Delta}(\Delta - \frac{x}{y})$, then the linear support function is

$$S(d,l) = \begin{cases} f(d,l) & \text{if } f(d,l) > 0 \\ 0 & otherwise \end{cases}$$

An outline of the algorithm is as follows :-

Algorithm 1: *Finding motion correspondence by support disparity without clustering.*

begin

F1 : = { X | X belongs to the first frame };
F2 : = { X | X belongs to the second frame };

(* Computing support for each disparity *)

for each element,p of F1 *do*
  for each element,q of F2 within a radius,R of p *do*
    Totalsupport(pq) : = 0;
    for each element,r of F1 *do*
      for each element,s of F2 within a radius,R of r *do*
        Support : = support of vector rs for vector pq
      end_for;
      Totalsupport(pq) : = Totalsupport(pq) + Support;
    end_for;
  end_for;
end_for;

(* Finding correspondences from the total supports *)

for each element p of F1 *do*
  for each element q of F2 within a radius,R of p *do*
    Find Maximum(Totalsupport(pq))
    (* the correspondence is given by
      vector pq = Maximum(Totalsupport(pq)) *)
  end_for;
end_for;

end. (* algorithm 1 *)

3.2. Motion detection by clustering in a space with reduced dimensions

The simple algorithm presented above works well in most instances. However, for cases where there are a large number of match possibilities for every point, the method is cumbersome. Thus in most cases in practical applications we need to add a reduced dimension clustering space in which implausible match possibilities are eliminated. If the match vectors are given by (x,y,ρ,θ) then the cluster space is obtained by dropping the spatial indexing coordinates (x,y). We then form clusters in the (ρ,θ) space. The hierarchical clustering process follows an algorithm given in [8]. The clustering metric is the Euclidean distance between two (ρ,θ) vectors. The number of clusters depends upon a threshold for the inter cluster distance. Let C_1 and C_2 be two clusters and $\mathbf{m}_1$ and $\mathbf{m}_2$ be the mean values for the respective cluster elements. In this case the inter cluster distance measure that we used was:

$$d = || \mathbf{m}_1 - \mathbf{m}_2 ||$$

The threshold is chosen depending on it stability, meaning that small changes to it should not affect the clustering in any significant way. The cluster trees (or dendograms) can be seen for some of the experimental data in figures 2.a.1 and 2.b.1.

The clusters so formed now compete against each other and only the larger clusters, i.e. the ones with accumulated votes in the same order of magnitude, are kept. These clusters then mediate the matching process in the lower level of displacement vectors. It is hoped that by this process two things are achieved:

1. Noise points and spurious matches are avoided.
2. In the case of multiple body motion, the clusters provide a convenient label for segmenting the displacement field.

An outline of the algorithm follows:

Algorithm 2: *Finding motion correspondence by clustering followed by application of support disparity.*

begin

FR1 : = {X | X is in the first frame }

FR2 : = {X | X is in the second frame }

(* setting up the table for clustering *)

for each element,p of F1 *do*
  for each element,q of F2 within a radius,R of p *do*
    disp_x : = (x coord. of q) - (x coord. of p);
    disp_y : = (y coord. of q) - (y coord. of p);
    clustertable[disp_x , disp_y] : =
      clustertable[disp_x , disp_y] + 1;
  end_for;
end_for;

Find the clusters in the two-dimensional array clustertable;
(* Refer figure 2.a.2 and 2.b.2 *)

Remove clusters with weak overall support (votes);
From the clusters find the feasible disparities;
Consider points in the feasible disparity ranges only, and apply Algorithm 1;

end.(*algorithm 2*)

The matching algorithm is formulated according to whether the points are labelled or not. In case of unlabelled points (as in the above algorithms) :

> All neighbouring points support (vote for) a particular disparity value. Similar values support each other strongly in a local region. Shorter length disparities are preferred. A point adopts a match for which it finds the maximum support.

The strategy is similar in spirit to the more sophisticated matchers, for instance, those using labelled points [23]. The feature points can carry labels which are computed from the outputs of the nine basis operators. A label is a code that identifies the image point in question. Now the matcher weights the "supporting" votes according to the similarity of these codes. However, we avoid iterative refinement, which is usually employed in similar algorithms [4].

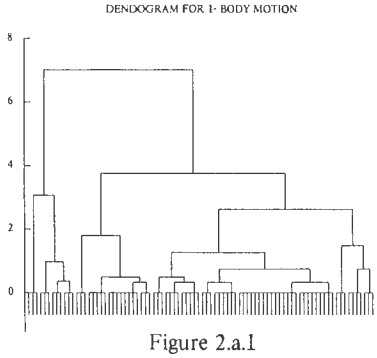


Figure 2.a.1

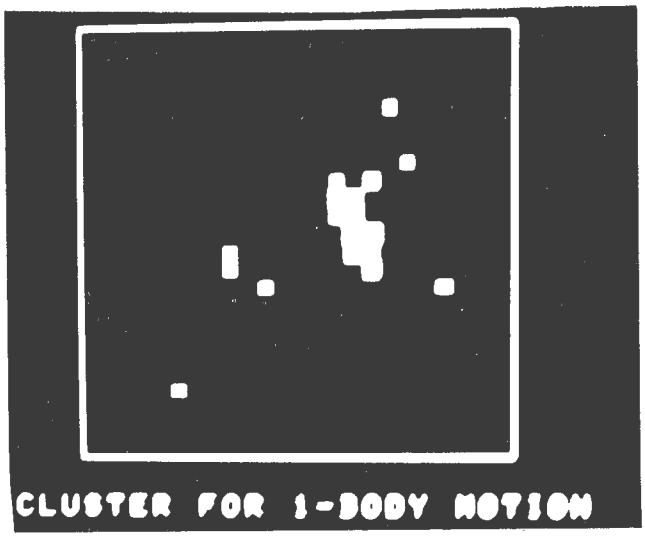


Figure 2.a.2

4. Experiments

Synthetic Images: Experiments have been conducted on synthetic images of spheres and planes "painted" with random dot patterns. All the objects are opaque and sometimes intersect each other. The choice of spheres and planes is motivated by the necessity of local smoothness in the imaged depth. Yet, at the same time, since there are multiple differently moving objects as well as occlusion of one body by another, motion bounderies do occur. The image formation technique was the perspective projection. In a single image there could be one or more instances of the above primitive objects moving with similar or different velocities (translational and rotational) in three space.

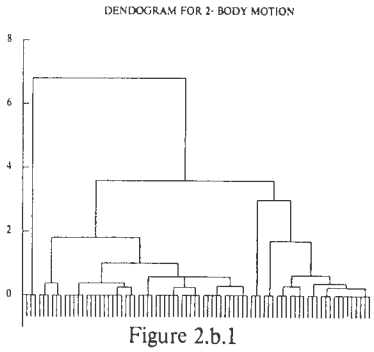


Figure 2.b.1

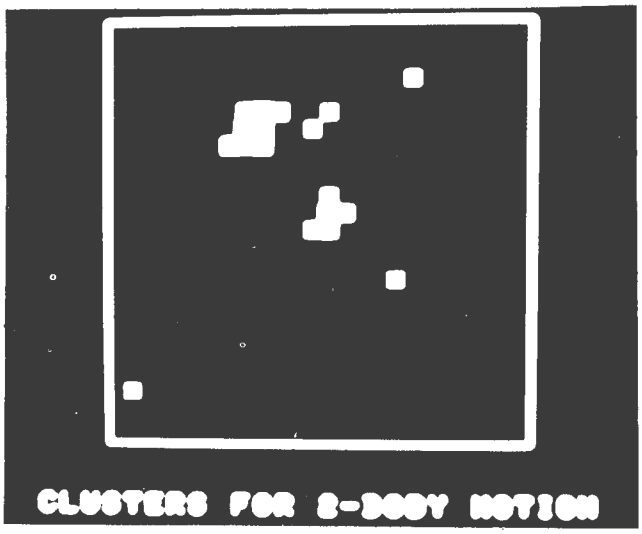


Figure 2.b.2

For single body motion the matches were found with close to 100% accuracy. Addition of uniformly distributed uncorrelated noise points to a level of 10% did not cause any significant difference in the level of correct matches found. However, the noise points generated some spurious matches among themselves. The clustering approach works better in this situation with considerable removal of noise points and false matches.

As a conservative estimate the average number of plausible match vectors considered was of the order of 10 - 15. Of course in regions with dense random dot patterns this number was more. Even with larger numbers the selection of the correct match was possible with the support disparity approach. These figures for plausible match vectors fell to a third of their number with the clustering approach. There was also a speedup of execution by a factor of around ten.

With two body motion about 94% of the correct matches could be obtained. The hardest matches to find lay on the border of the two bodies, as was to be expected. Thus on figure 3.1 and 3.2 we find that even though almost everywhere the matches have

been found correctly, on the border of the two bodies incorrect matches have been found. In the case of totally transparent bodies the algorithm's performance is drastically reduced with 50 -60 % wrong matches being found.

Images of Natural Scenes: Quantitative justification of performance is difficult on natural scenes. Through manual inspection it has been found that the number of wrong correspondences obtained are insignificant. However, correct matches associated with roughly 40% of the points have been found. This discrepancy is a result of the uncertainty associated with the determination of interest points.

It was computed that even with some amount of input/output processing, the part of the algorithm which could be parallelized took about 75% of the time on a serial processor. With the removal of file manipulations this could rise to over 95% or more of the time. It is feasible to implement the algorithm on a 128-processor MIMD machine (BBN Butterfly) with considerable improvement in running time.

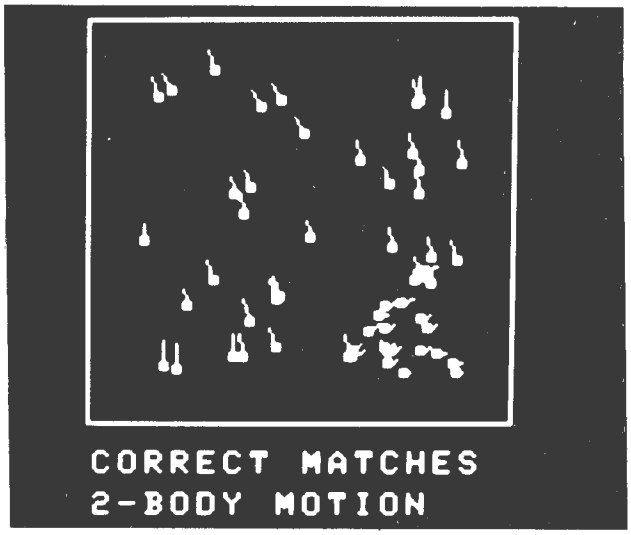


Figure 3.1

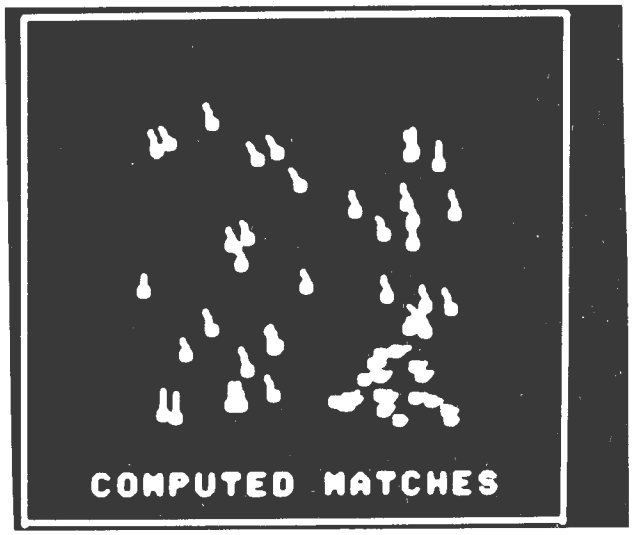


Figure 3.2

Figures 3.1 and 3.2 show the actual and the computed match vectors for synthetic images of two body motion.

5. Conclusion and Future Work

The goal of the work reported here, was to formulate a computational framework for the measurement of retinal motion. It was desirable that the motion measurement algorithm be implementable in parallel, and conform to a connectionist implementation strategy. An important consideration was graceful degradation in the presence of increasing amounts of noise, and the ability to handle multiple moving objects. An important issue, relating to the task of retinal motion measure is the choice of the matching primitive or token and the process for obtaining these primitives in an image. This issue was treated rather peripherally in this paper. This is because, the first priority was to devise a computational strategy based on the design criteria laid above. The overall framework of the algorithm is based on the matching paradigm of motion perception. This is built upon the belief that some form of matching, either involving spatio-temporal gradients or other feature primitives, is essential to solving the motion perception problem. This paradigm is by no means inviolable, as has been shown recently in [2], for certain imaging situations.

The work with synthetic images served to lay the preliminary groundwork for evaluating the proposed algorithm. The success in experiments with natural images is contingent upon being able to formulate and compute feature primitives that are stable and recoverable with local operators. The operator described previously (see also [3]) and used succesfully in other image processing contexts (see [1]) was used to test the algorithm on natural images. This "interest" operator lacks the sophistication of other corner finding algorithms like the ones described in [9, 16, 21]. However, it seems that more complicated interest operators do not fare too well than simple schemes, even in relatively uncomplicated scenes in nature (see [26]).

The two dimensional clustering approach of Algorithm 2 works well on most occasions. However for pathological cases like radial flow it is not at all advantageous to cluster in two dimensions because experimental observation with simulated data shows that no "good" cluster(s) is present in this case.

It is easy to circumvent the above problem by four dimensional clustering with the dimensions representing x , y, ρ , θ. However computationally there is not much advantage in using a four dimensional clustering approach over a direct application of Algorithm 1. The proper scheme to solve this problem is to start with two dimensional clustering and in the absence of "good" clusters to go in for a three dimensional clustering strategy with x ,y θ (θ is the orientation of the velocity vectors) and x , y, ρ (ρ is the magnitude of the velocity vectors). It is to be noted that in this case Euclidean distance is not a proper dissemblence measure. The support concept can be extended to get the dissemblence measure in this case. If "good" clusters are not found even after doing this Algorithm 1 is applied directly. For all the above stages the image is segmented into smaller divisions before the clustering process.

The above discussion should not convey the impression that clustering is an unsuitable method. Even in the worst case where no clustering is possible the application of algorithm 1 is not hindered in any way. However on most occasions as has been experimentally verified clustering is very useful in reducing the time taken for finding the correspondence.

To summarize, a list of the salient advantages of our algorithm is given below:

1. Applicable to multiple moving objects
2. Good behaviour in the presence of noise

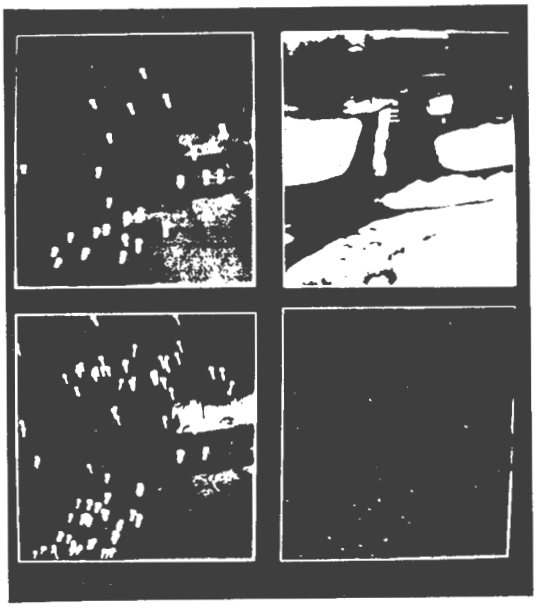


Figure 4:
Top Left : Manually computed vectors
Bottom Left : Computed vectors
Top Right : Image with enhanced contrast
Bottom Right: Interest Points

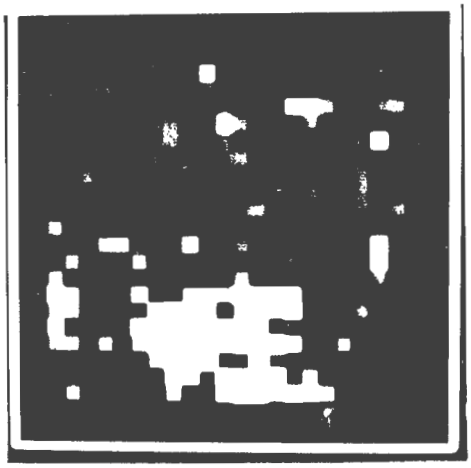


Figure 5: Cluster for Image of Fig. 4

3. Automatic segmentation for image areas projected from different moving objects or parts of the same rigid surface differentiated by sharp depth changes.
4. Conceptual simplicity and amenability to parallel implementation.
5. Adaptible to a different feature primitives and motion representations.

We are now working on the performance evaluation of the algorithm using several alternative feature detectors, as well as the alternative retinal motion parameter.

6. Acknowledgements

The preparation of this paper has been greatly aided by the critical comments of Chris Brown and Dana Ballard. The reviewers of this paper provided useful suggestions and comments. This work was supported by a National Science Foundation grant number DCR-8405720.

7. References

1. J. Aloimonos and M. J. Swain. Shape from Texture, *Proceedings of the IJCAI 2*, (August 1985), 926-931.
2. J. Aloimonos, A. Basu and C M. Brown, Contours, Shape, Motion, *DARPA IU workshop*, Miami, Florida. 1985.
3. A. Bandopadhay, Interest Points, Disparities and Correspondence, *DARPA Image Understanding Workshop*. , 1985.
4. S. T. Barnard and W. B. Thompson, Disparity Analysis of Images, *IEEE Trans. PAMI 2*, (1980), 333 - 340.
5. C. M. Brown, An iterative improvement algorithm for coherent codes, *Optics Comm.*, , June 1980.
6. C. M. Brown, M. B. Curtiss and D. B. Sher, Advanced Hough Transform Implementations, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, , 1983, 1081.
7. L. S. Davis, Z. Wu and H. Sun, Contour Based Motion Estimation, *CVGIP 23*, (1983), 313-326.
8. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
9. J. Q. Fang and T. S. Huang, A corner finding algorithm for Image Analysis, *Proc. AAAI-82*, Pittsburgh, Pennsylvania, August 1982, 46-49.
10. C. L. Fennema and W. B. Thompson, Velocity Determination in Scenes Containing Several Moving Objects, *CVGIP 9*, (1979), 301-315.
11. W. Frei and C. C. Chen, Fast boundery detection: A Generization and a New Algorithm, *IEEE Trans. Comput. 26*, (1977), 988 - 998.
12. R. M. Haralick and J. S. Lee, The Facet Approach to Optical Flow, *Proc. Image Understanding Workshop*, Arlington, Virginia, June, 1983, 84 - 93.
13. E. C. Hildreth, Computations Underlying the Measurement of Visual Motion, *Artificial Intelligence 23*, (1984), 309-354.
14. B. K. P. Horn and B. G. Schunck., Determining Optical Flow, *Artificial Intelligence 17*, (1981), 185-204.
15. M. H. Hueckel, An operator which locates edges in digital pictures, *J. ACM 18*, (1971), 113 - 125.
16. L. Kitchen and A. Rosenfeld, Gray - Level Corner Detection , Tech. Rep 887, Computer Science Center, University of Maryland, 1980.
17. D. Marr and E. Hildreth, Theory of Edge Dection, *Proc. Royal Soc. London. Series B. 207*, (1980), 187-217
18. D. Marr and S. Ullman, Directional Selectivity and its Use in Early Visual Processing, *Proc. Royal Soc. London. Ser. B 211*, (1981), 151-180.
19. D. Marr, *VISION*, W.H. Freeman, San Francisco, 1982.
20. H. P. Moravec, Towards Automatic Visual obstacle avoidance, *Proc. 5th IJCAI*, , 1977, 584.
21. H. Nagel, Displacement Vectors Derived from Second order Intensity Variations in Image Sequences, *CVGIP 21*, (1983), 85 - 117.
22. R. Paquin and E. Dubois, A Spatio-Temporal Gradient Method for Estimating the Displacement Field in Time Varying Imagery, *CVGIP 21*, (1983), 205-221.
23. J. M. Prager and M. A. Arbib, Computing The Optic Flow: The MATCH Algorithm and Prediction, *CVGIP 21*, (1983), 271-304..
24. K. Prazdny, Detection of Binocular Disparities, *Biol. Cybern. 52*, (1985), 93-99.
25. K. A. Stevens, Computation of Locally Parallel Structure, *Biol. Cybern. 29*, (1978), 19-28.
26. C. E. Thorpe, An Analysis of Interest Operators for FIDO, CMU-RI-Tech. Rep.-83-19, The Robotics Institute, Carnegie Mellon University, December 1983.
27. A. M. Waxman and K. Wohn, Contour evolution, neighbourhood deformation and global image flow:planar surfaces in motion, CAR-Tech. Rep.-74, Center for Automation Research, University of Maryland, April, 1984.

Determining the 3-D motion of a rigid surface patch without correspondence , under perspective projection .

John (Yiannis) Aloimonos and Isidore Rigoutsos
Department of Computer Science
The University of Rochester,
Rochester, New York 14627.

Abstract

A method is presented for the recovery of the 3D motion parameters of a rigidly moving textured planar surface. The novelty of the method is based on the fact that no point to point correspondences are used and that stereo and motion are combined in a way that does not require any static or dynamic point correspondences.

1. Introduction

An important problem in Computer Vision is to recover the 3-D motion of a mo- ving object from its successive images. Up to now there have been, basically, two aproaches towards the solution of this problem :

a) the differential or continuous

b) the discrete.

In the differential case, the dynamic image is supposed to be a 3-D function of two "spatial" arguments and a "temporal" argument. Then, if this function is locally well-behaved and its spatio-temporal gradients are computable, the local image velocity or "optical flow" may be computed [7,9,10,35].

The second approach, considers the cases where the motion is large and the first technique is not applicable. In these instances, the measurement of the image motion relies upon isolating and tracking highlights and feature points (corners, edges, etc) in the image through time. This entails tackling the correspondence problem which might be proved to be difficult in many situations [3,21,23,33,32].

In both of the above approaches, after the image motion (optical flow or discrete displacements) is computed, constraints are developed between the retinal motion and the 3-D motion parameters; these constraints become the basis of a whole variety of algorithms for the recovery of the 3-D motion [1,4,5,8,19,24,25, 27,28,29,30,37,38,36,18].

With the traditional one camera approach for the estimation of the 3-D motion parameters of a rigid planar patch, it was just mentioned [26],that one should use the image point correspondences for object points not on a single planar patch when estimating 3-D motions of rigid objects.

But it was not known, how many solutions there were, what was the minimum number of points and views needed to assure uniqueness and how could those solutions be computed without using any iterative search (i.e without having to solve non-linear systems). It was proved [27,28,30] that there are exactly two solutions for the 3-D motion parameters and plane orientations, given at least 4 image point correspondences in two perspective views, unless the 3x3 matrix containing the canonical coordinates of the second kind [20] for the Lie transfor- mation group that characterizes the retinal motion field of a moving planar patch, has multiple singular values.However, the solutions are unique if three views of the planar patch are given or two views with at least two planar patches. In our approach, the duality problem does not exist for two views, since two cameras are used (and so the analysis is done in 3 D).

2. Motivation and previous work

The basic motivation for this research is the fact that the optical flow (or discrete displacements) fields produced by existing methods applied to natural images, are corrupted by noise a great deal. In other words, despite the great mathematical elegance of the existing techniques for the computation of retinal motion, application of these methods to natural images does not give satisfactory results, because all of them are based on assumptions (this is absolutely necessary since the constraints for optical flow are not sufficient and so additional assumptions have to be employed), which hold true only for a subset of natural images [33,34].

However: "Are the existing algorithms or any other algorithms that are based on the optical flow input and one-camera and no other sources of information, robust enough to perform well with this kind of error, in their input ?". The answer seems to be "no", and this is probably due to the fact that the constraint relating the retinal motion to the 3-D motion (which is of course used by all algorithms involving the point-to-point correspondence approach) seems to be very sensitive to errors. Indeed, this seems to be the case as the following tables, from [30], show.

(The following Table 1. and Table 2. are from Tsai and Huang [30])

Error of point correspondences	0%	1%	3%	10%
Error of pure parameters	0%	4.10%	13.82%	85.84%
Error of rotation matrix	0%	1.74%	5.31%	19.11%
Error of translation	0%	5.88%	21.36%	275.20%

Table 1. effect of error of point correspondences on the error of motion parameters when 4-point correspondences are used, for the motion of a rigid planar patch. (using the algorithms in [30])

Number of point correspondences	4	5	10
Error of pure parameters	85.84%	12.57%	12.66%
Error of rotation matrix	19.11%	7.49%	6.57%
Error of translations	275.2%	14.12%	13.12%

Table 2. number of point correspondences as error of the estimated motion parameters with 10% error on point correspondences, for the motion of a rigid planar patch. (using the algorithms in [30])

We must admit that the results in [30], are from the best ones in the litterature and we use them here because they are from the few ones that include an error analysis.

Then a question arises, that is: "Is it possible to recover 3-D motion of a rigid planar patch without using any point correspondences and/or combining stereo with motion but without having to solve the stereo correspondence problem ? And in the

case this is possible, how robust against noise are the resulting algorith- ms ?". The need for such an approach has been realized lately [2,11,22].

However all the above methods require the derivatives of the image functions which in the case of noise are corrupted; some of them also involve the solutions of non-linear systems or knowledge of the environmental structure Finally, difficulties in the presence of noise are reported in [31] for the method in [2].

Also, recent work in that direction by Kanatani ([15,16]) describs two methods for the recovery of the 3-D motion of a moving planar contour without correspondence. Despite its elegance, the method is artificial and seems to suffer from numerical instabilities; furthermore, no behavior with respect to noise is presented. For the other approaches, robustness' behavior is not presented.

Finally, the need for combining stereo with motion has been realized by Huang and Blonstein ([12]) while it is worth mentioning that the need to combine stereo with motion at least for the computation of the retinal motion has been also appreciated by Jenkin and Tsotsos ([13]).

In this paper, we present a method for the recovery of the 3-D motion of a rigidly moving planar patch, by a binocular observer without using correspondence neither for the stereo nor for the motion.

3. Stereo without correspondence.

In this section we present a method for the recovery of the 3-D parameters for the set of 3-D planar points from their left and right images without using any point-to-point correspondence; instead we consider all point correspondences at once and so there is ne need to solve the difficult correspondence problem in the case of the static stereo

Let an orthogonal cartesian coordinate system OXYZ be fixed with respect to the left camera, with O at the origin (O being also the nodal point of the left eye) and the Z-axis pointing along the optical axis.

Let the image plane of the left camera be perpendicular to the Z-axis at the point (0,0,f), (focal length = f).

Let the nodal point of the right camera be at the point (d,0,0) and its image plane be identical to the left one; the optical axis of the right camera (eye) points also along the Z-axis and passes through point (d,0,0) (see **Figure 1.**).

Consider a set of 3-D points $A = \{ (X_i,Y_i,Z_i) / i=1,2,3 \dots n \}$ lying on the same plane(see **Figure 1.**), the latter being described by the equation :

$$Z = p*X + q*Y + c$$

Let O_l, O_r be the origins of the two-dimensional orthogonal coordinate systems on each image plane; these origins are located on the left and right optical axes while the corresponding coordinate systems have their y-axes parallel to the axis OY, and their x-axes parallel to OX.

Finally let $\{ (x_{li},y_{li}) / i=1,2,3 \dots n \}$ and $\{ (x_{ri},y_{ri}) / i=1,2,3 \dots n \}$ be the projections of the points of set A on the left and right retinae, respectively, i.e.

$$x_{li} = \frac{f*X_i}{Z_i} \quad (1) \qquad y_{li} = \frac{f*Y_i}{Z_i} \quad (2) \quad / \ i=1,2,3 \dots n$$

$$x_{ri} = \frac{f*(X_i - d)}{Z_i} \quad (3) \qquad y_{ri} = \frac{f*Y_i}{Z_i} \quad (4) \quad / \ i=1,2,3 \dots n$$

Let (x_{li},y_{li}) and (x_{ri},y_{ri}) be corresponding points in the two frames. Then we have that.

$$x_{li} - x_{ri} = \frac{f*d}{Z_i} \quad (5)$$

$$y_{li} = y_{ri} \quad (6)$$

where Z_i, the depth of the 3-D point having those projections.

It can be proved, [40], that the quantity

$$\sum_{i=1}^{n} \frac{y_{li}^k}{Z_i}$$

where

$$k \geq 0 \ \wedge \ k \neq \frac{m}{2*n}, \qquad m,n \in Z - \{0\},$$

is directly computable and equal to:

$$\sum_{i=1}^{n} \frac{y_{li}^k}{Z_i} = \sum_{i=1}^{n} \frac{x_{li} * y_{li}^k}{f*d} - \sum_{i=1}^{n} \frac{x_{ri} * y_{ri}^k}{f*d} \quad (7)$$

Note : In the case k >0, we require that k be a rational number with odd denominator so that the crresponding powers of the negative y_i's are defined.

3.1 Proposition : Using the aforementioned nomenclature, the parameters p, q and c of the plane in view are directly computable without using any point to point correspondence between the two frames

<Proof> The equation of the world plane when expressed in terms of the coordinates of the left frame, becomes :

$$\frac{1}{Z} = (f - p*x_l - q*y_l) * \frac{1}{c*f} \quad (8)$$

So, from equation (8) it follows that

$$\frac{1}{Z_i} = (f - p*x_{li} - q*y_{li}) * \frac{1}{c*f} \quad / \ i=1,2,3 \dots n \quad (9)$$

Now, we have :

$$\sum_{i=1}^{n} \frac{y_{li}^k}{Z_i} = \sum_{i=1}^{n} (f - p*x_{li} - q*y_{li}) * \frac{y_{li}^k}{c*f}$$

or

$$\sum_{i=1}^{n} \frac{y_{li}^k}{Z_i} = \frac{1}{c} * \sum_{i=1}^{n} y_{li}^k - \frac{1}{c*f} * [\sum_{i=1}^{n} p*x_{li}*y_{li}^k + \sum_{i=1}^{n} q*y_{li}*y_{li}^k] \quad (10)$$

The left-hand side of equation (10) has been shown to be computable without using any point-to-point correspondence (see Proposition 3.1).

If we write equation (10) for three different values of k, we obtain a **linear** system in the unknowns p,q,c which in general has a unique solution (except for the case where the projection of all points of set A, have the same y-coordinate in both frames).

$$\sum_{i=1}^{n} \frac{x_{li} * y_{li}^{k1}}{f*d} - \sum_{i=1}^{n} \frac{x_{ri} * y_{ri}^{k1}}{f*d} = \frac{1}{c} * \sum_{i=1}^{n} y_{li}^{k1} - \frac{1}{c*f} * [\sum_{i=1}^{n} p*x_{li}*y_{li}^{k1} + \sum_{i=1}^{n} q*y_{li}*y_{li}^{k1}] \quad (11)$$

$$\sum_{i=1}^{n} \frac{x_{li}^{*}y_{li}^{k2}}{f^{*}d} - \sum_{i=1}^{n} \frac{x_{ri}^{*}y_{ri}^{k2}}{f^{*}d} = \frac{1}{c} * \sum_{i=1}^{n} y_{li}^{k2} - \frac{1}{c^{*}f} * [\sum_{i=1}^{n} p^{*}x_{li}^{*}y_{li}^{k2} + \sum_{i=1}^{n} q^{*}y_{li}^{*}y_{li}^{k2}] \quad (12)$$

$$\sum_{i=1}^{n} \frac{x_{li}^{*}y_{li}^{k3}}{f^{*}d} - \sum_{i=1}^{n} \frac{x_{ri}^{*}y_{ri}^{k3}}{f^{*}d} = \frac{1}{c} * \sum_{i=1}^{n} y_{li}^{k3} - \frac{1}{c^{*}f} * [\sum_{i=1}^{n} p^{*}x_{li}^{*}y_{li}^{k3} + \sum_{i=1}^{n} q^{*}y_{li}^{*}y_{li}^{k3}] \quad (13)$$

The solution of this system recovers the structure and the depth of the points of set A without any correspondence and this is the conclusion of Proposition 3.1

3.3 Practical Considerations

We have implemented the above method for different values of k_1, k_2, k_3 and especially for the cases

a) $k_1 = 0$ $k_2 = 1/3$ $k_3 = 2/3$
b) $k_1 = 0$ $k_2 = 1/3$ $k_3 = 1/5$

The noiseless cases give extremely accurate results

Before we proceed, we must explain what we mean by noise introduced in the images. When we say that one frame (left or right) has noise of $a\%$, we mean that if the plane contains N projection points we added $[(N*a)/100]$ randomly distributed points. (*Note*: [] denotes the integer part of its argument)

When the noise in both frames is kept below 2% then the results are still very satisfactory. When the noise exceeds 5% then only the value of p gets corrupted, but the values of q and c remain very satisfactory. To correct this and get satisfactory results for high noise percentages, we devised the following method that uses three cameras :

" We consider the three camera configuration system as in **Figure 2.**, where the top camera has only vertical displacement with respect to the left one. If all three images are corrupted by noise (ranging from 5% to 20%) then application of the algorithm (**Proposition 3.1**) to the left and top frames will give very reasonable va-lues for p and c and corrupt q, which q, as well as c, are accurately computed from the application of the same algorithm to the right and left frames ".

4. Recovering the direction of translation

Here we treat the case where the points of set A just rigidly translate, and we wish to recover the direction of the translation. In this case, the depth is not needed but the orientation of the plane is required. The general case is treated in the next **Section 5.**

4.1 Technical framework.

Consider a coordinate system OXYZ fixed with respect to the camera; O coincides with the nodal point of the eye, while the image plane is perpendicular to the Z-axis (focal length = f), that is pointing along the optical axis (see **Figure 3.**).

Let us represent points on the image plane with small letters (e.g (x,y)) and points in the world with capital ones (e.g. (X,Y,Z)).

Let us consider a point $P=(X_1,Y_1,Z_1)$ in the world, with perspective image (x_1,y_1), where $x_1=(f*X_1)/Z$ and $y_1=(f*Y_1)/Z$.

If the point P moves to the position $P'=(X_2,Y_2,Z_2)$ with

$$X_2 = X_1 + \Delta X \quad (14)$$
$$Y_2 = Y_1 + \Delta Y \quad (15)$$
$$Z_2 = Z_1 + \Delta Z \quad (16)$$

then we desire to find the direction of the translation $(\Delta X/\Delta Z, \Delta Y/\Delta Z)$.

If the perspective image of P' is (x_2,y_2), then the observed motion of the world point in the image plane is given by the displacement vector : (x_2-x_1, y_2-y_1) (which in the case of very small motion is also known as "optical flow").

We can easily prove that :

$$x_2 - x_1 = \frac{f^{*}\Delta X - x_1^{*}\Delta Z}{Z_1 + \Delta Z} \quad (17)$$

$$y_2 - y_1 = \frac{f^{*}\Delta Y - y_1^{*}\Delta Z}{Z_1 + \Delta Z} \quad (18)$$

Under the assumption that the motion in depth is small with respect to the depth, the equations above become

$$x_2 - x_1 = \frac{f^{*}\Delta X - x_1^{*}\Delta Z}{Z_1} \quad (19)$$

$$y_2 - y_1 = \frac{f^{*}\Delta Y - y_1^{*}\Delta Z}{Z_1} \quad (20)$$

The above equations relate the retinal motion (left-hand sides) to the world motion $\Delta X, \Delta Y, \Delta Z$

4.2 Recovering the 3-D direction of translation without correspondence.

Consider again a coordinate system OXYZ fixed with respect to the camera as in **Figure 4.**, and let $A=\{ (X_i,Y_i,Z_i) / i=1,2,3 \ldots n \}$, such that

$$Z_i = p^{*}X_i + q^{*}Y_i + c \qquad / \ i=1,2,3 \ldots n$$

that is the points are planar. Let the points translate rigidly with translation $(\Delta X,\Delta Y,\Delta Z)$, and let $\{ (x_i,y_i) / i=1,2,3 \ldots n \}$ and $\{ (x_i',y_i') / i=1,2,3, \ldots n \}$ be the projections of the set A before and after the translation, respectively.

Consider a point (x_i,y_i) in the first frame which has a corresponding one (x_i',y_i') in the second (dynamic) frame

For the moment we do not worry about where the point (x_i', y_i') is, but we do know that the following relations hold between these two points :

$$x_i' - x_i = \frac{f^{*}\Delta X - x_i^{*}\Delta Z}{Z_i} \quad (21)$$

$$y_i' - y_i = \frac{f^{*}\Delta Y - y_i^{*}\Delta Z}{Z_i} \quad (22)$$

where Z_i is the depth of the 3-D point whose projection (on the first dynamic frame) is the point (x_i,y_i). Taking now into account that

$$\frac{1}{Z_i} = \frac{f - p^{*}x_i - q^{*}y_i}{c^{*}f} \quad (23)$$

the above equations become :

$$x_i' - x_i = (f^{*}\Delta X - x_i^{*}\Delta Z)^{*} \frac{f - p^{*}x_i - q^{*}y_i}{c^{*}f} \quad (24)$$

$$y_i' - y_i = (f^{*}\Delta Y - y_i^{*}\Delta Z)^{*} \frac{f - p^{*}x_i - q^{*}y_i}{c^{*}f} \quad (25)$$

If we now write equation (24) for all the points in the two dynamic frames and sum the resulting equations up, we take:

$$\sum_{i=1}^{n}(x_i'-x_i)=\sum_{i=1}^{n}[(f^*\Delta X-x_i^*\Delta Z)^*\frac{f-p^*x_i-q^*y_i}{c^*f}]$$

or

$$\sum_{i=1}^{n}(x_i'-x_i)=\sum_{i=1}^{n}[\frac{f^*(f-p^*x_i-q^*y_i)^*\Delta X-x_i^*(f-p^*x_i-q^*y_i)^*\Delta Z}{c^*f}] \quad (26)$$

Similarly, if we do the same for equation (25), we take :

$$\sum_{i=1}^{n}(y_i'-y_i)=\sum_{i=1}^{n}[(f^*\Delta Y-y_i^*\Delta Z)^*\frac{f-p^*x_i-q^*y_i}{c^*f}]$$

or

$$\sum_{i=1}^{n}(y_i'-y_i)=\sum_{i=1}^{n}[\frac{f^*(f-p^*x_i-q^*y_i)^*\Delta Y-y_i^*(f-p^*x_i-q^*y_i)^*\Delta Z}{c^*f}] \quad (27)$$

At this point it has to be understood that equations (26) and (27) do not require our finding of any correspondence.

By dividing equation (26) by equation (27), we get

$$\frac{\sum_{i=1}^{n}x_i'-\sum_{i=1}^{n}x_i}{\sum_{i=1}^{n}y_i'-\sum_{i=1}^{n}y_i}=\frac{\sum_{i=1}^{n}[\frac{\Delta X}{\Delta Z}^*f^*(f-p^*x_{1i}-q^*y_{1i})-(f-p^*x_{1i}-q^*y_{1i})^*x_{1i}]}{\sum_{i=1}^{n}[\frac{\Delta Y}{\Delta Z}^*f^*(f-p^*x_{1i}-q^*y_{1i})-(f-p^*x_{1i}-q^*y_{1i})^*y_{1i}]} \quad (28)$$

Equation (28) is a linear equation in the unknowns $\Delta X/\Delta Z$, $\Delta Y/\Delta Z$ and the coefficients consist of expressions involving summations of point coordinates in both dynamic frames; for the computation of the latter no establishment of any point correspondences is required. It is clear, of course, that the parameters used in this formula are the ones computed using xthe method described in **Section 3**. Furthermore, it has to be pointed out that, as the experiments showed, errors of even up to 30% in the values of p,q had insignificant influence to the computation of the two unknowns $\Delta X/\Delta Z$, $\Delta Y/\Delta Z$.

So, if we consider a binocular observer, applying the above procedure in both left and right "eyes", we get two linear equations (of the form of equation (28)) in the two unknowns $\Delta X/\Delta Z$, $\Delta Y/\Delta Z$, which constitute a linear system that in general has a unique solution.

4.3 Practical considerations.

We have implemented the above method with a variety of planes as well as displacements. noiseless cases give extremely accurate results, while cases with noise percentages up to 20% (even with different amounts of noise in all four frames (first left and right - second left and right)) give very satisfactory results (an error of at most 5%) . **Section 6** describes relevant experiments. We now proceed considering the general case.

5. Determining unrestricted 3-D motion of a rigid planar patch without correspondence.

Consider again the imaging system (binocular) of **Figure 4.**, as well as the set $A=\{(X_i,Y_i,Z_i)/i=1,2,3\ldots n\}$ such that :

$Z_i=p^*X_i+q^*Y_i+c \quad / \quad i=1,2,3\ldots n$

i.e. the points are planar; let B be the plane on which they lie. Suppose that the points of the set A move rigidly in space (translation plus rotation) and they become members of a set $A'=\{(X_i',Y_i',Z_i')/i=1,2,3\ldots n\}$. Since all of the points of set A move rigidly, it follows that the points of set A' are also planar; let B' be the (new) plane on which these points lie.

In other words the set A becomes A' after the rigid motion transformation. We wish to recover the parameters of this transformation . From the projection of sets A and A' on the left and right image planes and using the method described in **Section 3** the sets A and A' can be computed. In other words, we know exactly the positions in 3-D of all the points of the sets A and A' (and this has been found without using any point correspondences - **Section 3**).

So, the problem of recovering the 3-D motion has been transformed to the following

"Given the set A of planar points in 3D and the set A' of new planar points, which has been produced by applying to the points of set A a rigid motion transformation, recover that transformation."

Any rigid body motion can be analyzed to a rotation plus a translation; the rotation axis can be considered as passing through any point in the space, but after this point is chosen, everything else is fixed.

If we consider the rotation axis as passing through the center of mass (CM) of the points of set A, then the vector which has as its two endpoints the centers of mass $\mathbf{CM_A}$ and $\mathbf{CM_{A'}}$ of sets A and A' respectively, represents the *exact* 3-D translation.

So, for the translation we can write

$$\textbf{translation} \equiv \mathbf{T}=(X,Y,Z)=\mathbf{CM_{A'}}-\mathbf{CM_A}$$

It remains to recover the rotation matrix.

Let, therefore, $\mathbf{n_1}$ and $\mathbf{n_2}$ be the surface normals of the planes B and B'. Then, the angle θ between $\mathbf{n_1}$ and $\mathbf{n_2}$, where

$$\cos\theta=\frac{\mathbf{n_1}\cdot\mathbf{n_2}}{\|\mathbf{n_1}\|^*\|\mathbf{n_2}\|} \quad , \text{ with } '\cdot' \text{ the inner-product operator}$$

represents the rotation around an axis O_1O_2 perpendicular to the plane defined by $\mathbf{n_1}$ and $\mathbf{n_2}$, where

$$\mathbf{O_1O_2}=\frac{\mathbf{n_1}\times\mathbf{n_2}}{\|\mathbf{n_1}\times\mathbf{n_2}\|} \quad , \text{with } '\times' \text{ the cross-product operator}$$

From the axis O_1O_2 and the angle θ we develope a rotation matrix $\mathbf{R_1}$. The matrix $\mathbf{R_1}$ does not represent the final rotation matrix since we are still missing the rotation around the surface normal. Indeed, if we apply the rotation matrix $\mathbf{R_1}$ and the translation **T** to the set A, we will get a set A" of points, which is different than A', because the rotation matrix $\mathbf{R_1}$ does not include the rotation around the surface normal $\mathbf{n_2}$.

So we now have a matching problem : on the plane B' we have two sets of points A' and A" respectively, and we want to recover the angle ϕ by which we must rotate the points of set A" (with respect to the surface normal $\mathbf{n_2}$) in order to coincide with those of set A'.

Suppose that we can find angle ϕ. From ϕ and n2 we construct a new rotation matrix $\mathbf{R_2}$. The final rotation matrix **R** can be expressed in terms of $\mathbf{R_1}$, $\mathbf{R_2}$ as follows :

$$\mathbf{R}=\mathbf{R_1}\mathbf{R_2}$$

It therefore remains to explain how we can compute the angle ϕ. For this we need the statistical definition of the mean direction.

Definition 1 . Consider a set $S=\{(X_i,Y_i)/i=1,2,3\ldots n\}$ of points all of which lie on the same plane. Consider the center of mass, CM, of these points to have coordinates (X_{cm},Y_{cm}). Let also circle (CM,1) be the circle having its center at (X_{cm},Y_{cm}) and radius of length equal to 1. Let P_i be the intersections of the vectors $\mathbf{CMS_i}$ with the circumference of the circle (CM,1), $i=1,2,3\ldots n$. Then the "mean direction" of the points of the set A, is defined to be the vector **MD**, where

$$\mathbf{MD}\equiv\sum_{j=1}^{n}\mathbf{CMP_j}$$

It is clear that the vector of the mean direction is intrinsically connected with the set of points considered each time, and if the set of points is rotated around an axis perpendicular to the plane and passing through CM, by an angle ω, the new mean direction vector is the previous one rotated by the same angle ω.

So, returning to the analysis of our approach, the angle ϕ is the angle between the vectors of mean directions of the sets A' and A" (which have obviously, common CM's). recall at this point that A' and A" are sets of coplanar points; therefore we can apply the above definition.

Moreover, it is obvious that the angle ϕ, and therefore the

rotation matrix $\mathbf{R_2}$, cannot be computed in the case the mean direction is **0** (i.e. in the case the set of points is characterized by a point symmetry).

6.Experiments .

We will describe experiments for both the detection of structure and depth without correspondence and the detection of 3-D motion without correspondence.

In our experiments, we considered a set of three-dimensional planar points, which we projected perspectively in both the left and right frames. From the projections we recover the structure and depth of the 3-D plane using the alogrithm described in Section 3 , or using the projections in three frames . It is clear, that the equations that are used to develop the linear system described in **Section 3,** are based on the assumption that the number of points on (left and right frames), is the same. But in noisy situations, this is not the case In particular, in real images operators have first to be applied on all four frames (two before the motion and two after the motion) that will produce points of interest, ([3,6,17,21]) and then the theory developed in this paper is applied to these points

But any method that will produce points of interest from intensity images is bound to have errors due to the noise in the images and the unpredictable behavior of the intensity function in natural scenes When we say that the methods that find intere sting points in intensity images are bound to errors, we mean that there will be points in the left frame whose corresponding ones have not been found in the right stereo frame, and also there will be points in the first dynamic frame whose cor responding ones have not been found in the second dynamic frame, and vice versa. So, the number of points will not be the same in the different images . Because of that, our method is bound to have an error, since it is based on the assumption that the number of points is everywhere the same. To reduce this error we do the follow- ing: Equations (11), (12), (13) are not affected if both sides are divided by the number of points in all the frames (under the assumption that the number of points is the same in all frames). If now the numbers of points in the left and right fram e are different, say n_{left} and n_{right} , in the static stereo case, then we divide the summations resulting from each of the frames, by the number of points of the corresponding frame, and the resulting equations are (for the static stereo case):

$$\sum_{i=1}^{n}\frac{x_{li}*y_{li}^{k1}}{f*d*n_{left}}-\sum_{i=1}^{n}\frac{x_{ri}*y_{ri}^{k1}}{f*d*n_{right}}=\frac{1}{c*n_{left}}*\sum_{i=1}^{n}y_{li}^{k1}-\frac{1}{c*f*n_{left}}*[\sum_{i=1}^{n}p*x_{li}*y_{li}^{k1}+\sum_{i=1}^{n}q*y_{li}*y_{li}^{k1}] \quad (29)$$

$$\sum_{i=1}^{n}\frac{x_{li}*y_{li}^{k2}}{f*d*n_{left}}-\sum_{i=1}^{n}\frac{x_{ri}*y_{ri}^{k2}}{f*d*n_{right}}=\frac{1}{c*n_{left}}*\sum_{i=1}^{n}y_{li}^{k2}-\frac{1}{c*f*n_{left}}*[\sum_{i=1}^{n}p*x_{li}*y_{li}^{k2}+\sum_{i=1}^{n}q*y_{li}*y_{li}^{k2}] \quad (30)$$

$$\sum_{i=1}^{n}\frac{x_{li}*y_{li}^{k3}}{f*d*n_{left}}-\sum_{i=1}^{n}\frac{x_{ri}*y_{ri}^{k3}}{f*d*n_{right}}=\frac{1}{c*n_{left}}*\sum_{i=1}^{n}y_{li}^{k3}-\frac{1}{c*f*n_{left}}*[\sum_{i=1}^{n}p*x_{li}*y_{li}^{k3}+\sum_{i=1}^{n}q*y_{li}*y_{li}^{k3}] \quad (31)$$

where n_{left} and n_{right} represent the numbers of points in the left and right frames respectively. It is clear that the resulting equations are approximate, but our experi- ments show that the introduced error is very small. It has to be mentioned , however, that the intrinsic difficulty, appearing in the traditional methods (i.e. stereo, optical flow), of not being able to find corresponding points, exists even in our algorithm but under the form of different numbers of points in the different frames, because of the globality of our approach. However, even considerable differences in the numbers of points among the different frames hardly affects the results. Furthermore, the same technique is applied to the case of motion as well.

Picture 1. shows the projections of a set of planar points on both the left and right frames. The frame on top is the superposition of the left and right frames. The actual parameters of the plane were : p = 0.0 , q = 0.0 , c = 10000, while the number of points was equal to 1000.We did not include any noise to our pictures.
The computed ones were : P = -0.0 , Q = -0.0 , C = 10000.0

Picture 2. shows the projections of a set of planar points on both the left and right frames. The frame on top is the superposition of the left and right frames. The actual parameters of the plane were : p = 1.0 , q = 1.0 , c = 10000, while the number of points was equal to 1000.We did not include any noise to our pictures.
The computed ones were: P = 0.98 , Q = 1.00 , C = 9809.8

Picture 3. shows the projections of a set of planar points on both the left and right frames. The frame on top is the superposition of the left and right frames. The actual parameters of the plane were : p = 1.0 , q = 1.0 , c = 10000, while the number of points was equal to 1000.We included 5% noise to the left frame and 7% to the right one. The computed ones were P = 1.7 , Q = 1.2 , C = 10266.7

Pictures 4a., 4b. show the results from the *3-eye method*. Here the projections of a set of 3-D planar points on all the three frames are considered. The actual parameters were:p = 0.0 , q = 0.0 , c = 10000 (**Picture 4a.**) and p = 1.50 , q = 2.30 , c = 10000 (**Picture 4b.**) respectively. The number of points was equal to 1000, in both pictures.

Picture 4b. did not have any noise, whereas **Picture 4a.** had 5% noise in the left frame and 7% noise in the right and top frames.
The computed ones were. P = 0.10 , Q = 0.05 , C = 10197.0 and P = 1.51 , Q = 2.22 , C = 10000.0 respectively

Pictures 5.,6.,7.,8.,9., show the 3-D motion determination results. In **Picture 5.,** the two frames at the bottom represent the projections of a set of 3-D planar points on the left and right eyes respectively. The two frames at the top, represent the projections of the same set of points, after it has been translated The actual direc- tion of translation was equal to (-2.0 , 2.0) , and the computed one was (-1.9, 2.0).

The noise percentage was equal to 10% in all four frames while the number of points was equal to 1000. At this point it has to be mentioned that the parameters p,q were also computed computed,since the latter are used in the determination of the direction of translation (see also **Section 4**).**Pictures 6.,7.,** represent similar experiments **Pictures 8.** and **9.** , show experiments determining the general motion The results were computed according to the method presented in **Section 5.**, and the results were recalculated with respect to the left-camera coordinate system.

NOTE: All the parameters involved in the above experiments that have a dimension of length ($L^1 M^0 T^0$) are calculated in pixels , where 1 pixel = 100μm.

7. CONCLUSION AND FUTURE WORK

We have presented a method on how a binocular (or trinocular) observer can recover the structure, depth, and 3-D motion of rigidly moving planar patch without using any static or dynamic point correspondences. We are currently setting up the the experiment for the application of the method in natural images. The theory for the recovery of unrestricted 3-D motion without using point correspondences has been extended for the case of non-planar surfaces. The interested reader can consult [40].

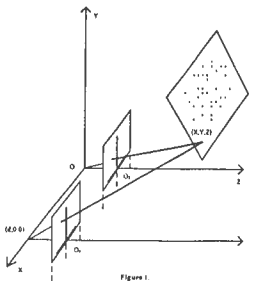
8. ACKNOWLEDGMENTS

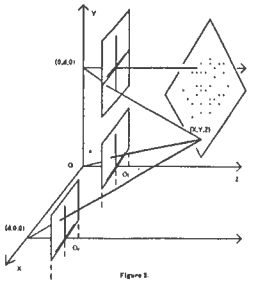
This work was supported by a research contract from the U.S. Army Engineer Topographic Laboratories (Number DACA 76-85-0001).

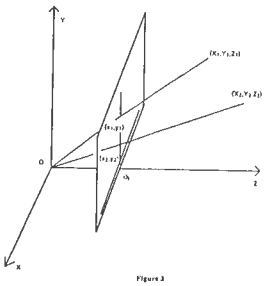
REFERENCES

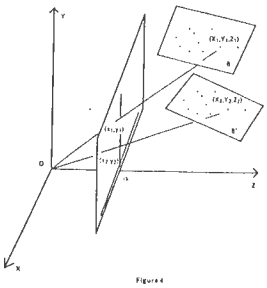
1. G. Adiv , Determining 3-D Motion and Structure from Optical Flow Generated from Several Moving Objects COINS-Tech. Rep. 84-07, July 1984.
2. J. Aloimonos, and C. M. Brown, Direct Processing of Curvilinear Sensor Motion from a Sequence of Perspective Images, Proc. of the IEEE Workshop on Computer Vision, Representation and Control, Annapolis, Maryland,1984.
3. A. Bandyopadhyay A Multiple Channel Model for the Perception of Optical Flow, IEEE Proc. of the Workshop on Computer Vision Representation and Control,1984, pp78-82.
4. A. Bandyopadhyay and J. Aloimonos, Perception of Rigid Motion from Spatio-Temporal Derivatives of Optical Flow, Tech. Report 157, Dept. of Computer Science, University of Rochester, 1985.
5. A. Bruss and B.K.P. Horn, Passive Navigation, CVGIP 21, 1983, pp3-20
6. S.T. Barnard and W.B. Thompson, Disparity analysis of images, IEEE Trans PAMI, Vol. 2, 1980, pp 333-340.
7. L.S. Davis, Z. Wu and H. Sun, Contour Based Motion Estimation, CVGIP 23 1983, pp313-326
8. J.Q. Fang and T.S. Huang, Solving Three Dimensional Small Rotation Motion Equations: Uniqueness, Algorithms and Numerical Results, CVGIP 26, 1984 pp183-206.

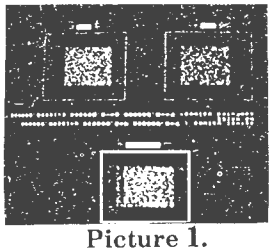
9. R M. Haralick and J.S.Lee, The Facet Approach to Optical Flow, Proc Image Understanding Workshop, Arlington, Virginia, June, 1983. pp84 93
10. B.K.P. Horn and B.G. Schunck, Determining Optical Flow. Artificial Intelligence 17, 1981, pp185- 204.
11 T.S. Huang, Three Dimensional Motion Analysis by Direct Matching, Topical Meeting on Machine Vision, Incline Village, Nevada, 1985, ppFA1.1 - FA1.4.
12. T.S. Huang and S.D. Blonstein, Robust Algorithms for Motion Estimation Based on Two Sequential Stereo Image Pairs, Proc. CVPR 1985, pp518-523.
13. M. Jenkin and J. Tsotsos, Applying Temporal Constraints to the Dynamic Stereo Problem, Department of Computer Science, University of Toronto, CVGIP.
14. T. Kanade, Camera Motion from Image Differentials, Proc. Annual Meeting, Optical Society of America, Lake Tahoe, March 1985.
15. K.-I. Kanatani, Detecting the Motion of a Planar Surface by Line and Surface Integrals, CVGIP, Vol. 29, 1985, pp13-22
16. K.-I. Kanatani, Tracing Planar Surface Motion from a Projection without Knowing the Correspondence, CVGIP, Vol 29, 1985, pp1-12.
17. L, Kitchen and A. Rosenfeld, Gray Level Corner Detection, TR 887, Computer Science Dept., U. of Maryland, 1980
18. H.C. Longuet-Higgins, A Computer Algorithm for Reconstructing a Scene from 2 Projections, Nature 293, 1981.
19. H.C.Longuet-Higgins and K.Prazdny, The Interpretation of a Moving Retinal Image, Proc. Royal Society London B 208, 1980, pp385 397.
20. Y. Matsushima, Differentiable Manifolds, New York Marcel Dekker, 1972.
21. H.P.Moravec, Towards Automatic Visual Obstacle Avoidance, Proc., 5th IJCAI, 1977.
21. H. Nagel, Displacement Vectors Derived from Second Order Intensity Variations in Image Sequences, CVGIP, Vol 21, 1983, pp85 117.
22. S. Negahdaripour and B.K Horn, Determining 3-D Motion of Planar Objects from Image Brightness Patterns, IJCAI85, 1985, pp898-901
23. J.M. Prager and M.A. Arbib, Computing the Optical Flow : The MATCH Algorithm and Prediction, CVGIP, Vol. 21, 1983, pp272-304.
24. K.Prazdny, Determining the Instantaneous Direction of Motion from Optical Flow Generated by Curvilinearly Moving Observer, CVGIP, Vol. 17, 1981, pp 94 97.
25. J.H.Rieger and D.T Lawton, Determining the Instantaneous Axis of Translation from Optic Flow Generated by Arbitrary Sensor Motion, COINS Tech. Report 83-1., January 1983.
26. J.W. Roach and J.K. Aggarwal, Determining the Movement of Objects from a Sequence of Images, PAMI, Vol. 2, No. 6, November 1980, pp554-562.
27. R.Y.Tsai, T.S.Huang and W. Zhu, Estimating Three Dimensional Motion Parameters of a Rigid Planar Patch II: Singular Value Decomposition, IEEE Trans. A.S.S.P. ASSP-30, August 1982, pp525-534
28. R.Y Tsai, T.S.Huang, Estimating Three Dimensional Motion Parameters of a Rigid Planar Patch III: Finite Point Correspondences and The Three View Problem, Proc. IEEE Conf. ASSP Paris, May 1982.
29. R Y Tsai, T.S.Huang, Uniqueness and Estimation of Three Dimensional Motion Parameters of Rigid Objects with Curved Surfaces, IEEE. Trans. PAMI 6, January 1984, pp13-27.
30. R Y Tsai, T.S.Huang, Uniqueness and Estimation of Three Dimensional Motion Parameters of Rigid Objects, Image Understanding 1984, eds Shimmon Ullman and Whitman Richards, Ablex Publ. Co., New Jersey. 1984, pp135
31 B. Sarachan, Experiments in Rotational Egomotion Calculation, TR 152 Computer Science Dept., U. of Rochester, 1985
32. S Ullman, The Interpretation of Visual Motion, Ph.D Thesis, 1977.
33. S. Ullman Analysis of Visual Motion by Biological and Computer Systems, IEEE Computer, 14 (8), 1981, pp57-69
34 S. Ullman, Computational Studies in the Interpretation of Structure and Motion: Summary and Extension, AI Memo No 706, Mit AI Lab., March 1983.
35. S. Ullman and E. Hildreth, The Measurement of Visual Motion, in Physical and Biological Processing of Images (Proc. Int. Symp. Rank Prize Funds London), O.J. Braddick and A.C. Sleigh (ed.), Springer Verlag, September 1982, pp154- 176.
36. B. Yen and T.S. Huang, Determining 3-D Motion and Structure of Rigid Objects Containing Lines Using the Spherical Projection, in Image Sequence Processin& Dynamic Scene Analysis, T.S. Huang (ed.), 1983.
37. A.M. Waxman and S.S. Sinha, Dynamic Stereo: Passive Ranging to Moving Objects from Relative Image Flows, CAR-Tech. Report-74, College Park, Mary land 20742, July 1984.
38. A. Waxman and S. Ullman, Surface Structure and 3-D Motion from Image Flow: A Kinematic Approach, CAR-TR-24, Center for Automation Research, Univ. of Maryland, 1983.
39. K. Wohn and A.Waxman, Contour Evolution, Neighbourhood Deformation and Local Image Flow: Curved Surfacesin Motion, CAR-TR-134, Computer Vision Lab., Univ. of Maryland.
40. J. Aloimonos and I. Rigoutsos,"Determining the 3-D motion of a moving rigid surface patchwithout correspondence", TR 178, Dept. of Computer Science University of Rochester, Rochester, N.Y. 14627.



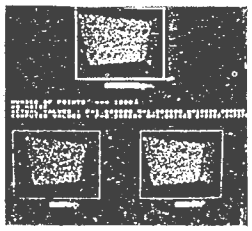




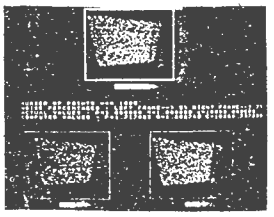




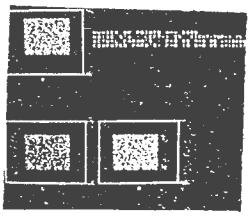
Picture 1.



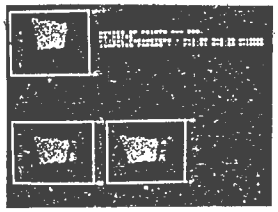
Picture 2.



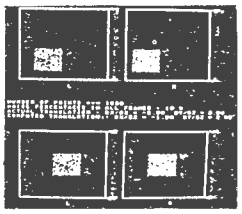
Picture 3.



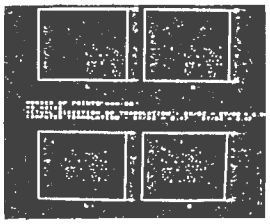
Picture 4a.



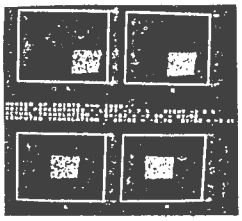
Picture 4b.



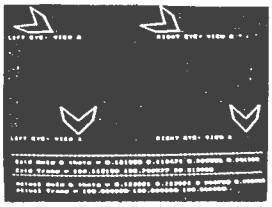
Picture 5.



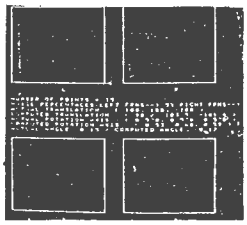
Picture 6.



Picture 7.



Picture 9.



Picture 8.

Active Navigation

Amit Bandopadhay, Barun Chandra
and Dana H. Ballard

Department of Computer Science
The University of Rochester
Rochester, NY 14627

ABSTRACT

The visual navigation problem deals with the computation of egomotion parameters of a moving observer based on visual stimulus. The term Active Navigation refers to the fact that the observer actively tracks environmental points. Conventional methods for the computation of the rigid motion parameters from retinal stimulus suffer from a parameter space of high dimensionality. This difficulty is compounded by the nonlinearity of the constraint equations that govern the computation of the motion parameters from monocular retinal flow. The technique proposed is free from the above difficulties.

1. Introduction

The two most serious barriers to the computation of the parameters of rigid motion from the retinal optical flow are:

(a) High dimensionality of the parameter space.

(b) The nonlinearity of the constraint equations.

This paper suggests that these handicaps may be lessened by a twofold strategy. The most important of these is to employ tracking. The other is to employ stereoscopic imaging.

The scenario of interest in this paper concerns the observer moving in a static environment. The problem addressed here analyses the navigational capability of the observer based on visual stimulus alone. In other words the question to be answered is how well the observer can recover Egomotion parameters from retinal motion stimulus. In general, motion perception involves the recovery of the parameters of motion, as well as the structure (or shape) of the object(s) in view. The geometric properties of the three dimensional surfaces and points are related to the geometry of their image.

Thus the projective transformation involved in the image formation process must be analyzed to find out how the retinal stimulus constrains the three dimensional motion parameters.

To begin with, the input to the motion interpretation process is assumed to be the optical flow induced on the retina by the reflected light pattern from environmental points. The measurement of this image motion is itself a topic of intense study in the last few years [5, 6, 11, 12, 18]. The difficulty of optical flow measurement stems from the fact that local measurement of the flow field entails solving the "correspondence problem" [2, 11]. This involves matching some kind of feature points across different time frames of a dynamically imaged scene - a problem that has not been resolved satisfactorily as yet. However in spite of these difficulties, for the moment, the stimulus will be assumed to be optical flow since:

(a) Most importantly, it is mathematically, a convenient representation which clearly encapsulates the constraint that image motion brings to bear on the rigid motion parameters.

(b) The smoothness property of optical flow is likely to help reduce noise in the input to the motion estimation process and help segmentation in dynamic images [7, 19].

(c) Extensive analyses of the constraints imposed by the optical flow field upon the rigid motion parameters have been performed [1, 4, 8, 10, 17].

In the case when the observer is able to track a prominent feature point in the imaged scene, then the observer's task of navigation is facilitated since it is easier to compute his Egomotion parameters, compared to the non-tracking case. This paradigm of "Active Navigation" is investigated both for a monocular as well as a binocular observer.

The contributions of the research, reported in this paper are summarized as follows:

1. A general form of the relation between 3D velocity parameters and retinal optical flow is derived. In previous derivations of this relation [8] the origins of the body centered coordinate frame and viewer centered coordinate frame are taken to coincide at the instant of measurement. Using the general representation it is shown why a monocular observer, who is able to track an environmental feature point, has to contend with a smaller number

of velocity parameters.

2. The analysis extends naturally to stereo imaging situations, where it is shown that, by combining measurements from both eyes, a linear equation in two unknowns is obtained.
3. The above constraint is applied with all possible stereo correspondences in a small neighbourhood, so as to minimise the square error. This least square error technique is seen to work well on simulated data, even with the addition of 10-20 percent noise.

2. Mathematical formulation of the problem

Consider first, the monocular imaging situation where a sensor is moving relative to a static scene. The co-ordinate frame (X,Y,Z) is fixed to the sensor (see figure I). The viewing direction is along the positive Z-axis.

A rigid body is defined as a set of points whose relative euclidian distances from all other points in the set are invariants with respect to the transformations of rotation and translation. In addition, only opaque objects and hence points on a surface (or a manifold) in three space will be considered. In other words the three cartesian coordinates of of a point on a rigid body are not independent.

The problem deals with a rotating and translating observer moving in a static environment. The analysis uses the velocity representation for the motion parameters.

The reference coordinate frame is fixed to the observer. There is another coordinate frame fixed at the point 'S' on the body. The point S has the velocity $T_s = (U_s, V_s, W_s)$. At the time of observation the reference and the body frame axes are parallel to each other. The rotational velocity of the body is given by the vector $\Omega = (\alpha, \beta, \gamma)$. The 3D velocity of a point $P = (X, Y, Z)$ on the body is given by the equation

$$\dot{\mathbf{X}} = \mathbf{T} + [R](\mathbf{X} - \mathbf{X_s}) \qquad (2.1)$$

where $\mathbf{X_s} = (X_s, Y_s, Z_s)$ denote the position of the body origin 'S' , and $\dot{\mathbf{X}}$ denotes the 3D velocity of P (the 'dot' operator is used throughout to signify differentiation with respect to time), also

$$[R] = \begin{bmatrix} 0 & -\gamma & \beta \\ \gamma & 0 & -\alpha \\ -\beta & \alpha & 0 \end{bmatrix}$$

The image formation is modeled by the perspective projection model. The projection of a point $P=(X,Y,Z)$ is denoted by $p=(x,y)$. The projective relation is

$$(x,y) = \left(\frac{FX}{Z}, \frac{FY}{Z}\right) \qquad (2.2)$$

The constant F is the focal length of the imaging system. It is the distance separating the nodal point of the camera (or eye) and the image plane, moving along the optical axis (i.e. Z axis). In subsequent steps the constant F is assumed to be unity. The velocity of image points in the 2d image space is called optical flow. The relations between the 2D and 3D velocities are obtained by differentiating the equation (2.2) and substituting from equation (2.1).

$$\begin{aligned} u = \dot{x} &= \frac{U_s - xW_s}{Z} - \alpha\left[xy - x\frac{Y_s}{Z}\right] \\ &+ \beta\left[1 - \frac{Z_s}{Z} + x^2 - x\frac{X_s}{Z}\right] - \gamma\left[y - \frac{Y_s}{Z}\right] \\ v = \dot{y} &= \frac{V_s - yW_s}{Z} - \alpha\left[1 - \frac{Z_s}{Z} + y^2 - y\frac{Y_s}{Z}\right] \\ &+ \beta\left[xy - y\frac{X_s}{Z}\right] + \gamma\left[x - \frac{X_s}{Z}\right] \end{aligned} \qquad (2.3)$$

When the origin of the body coordinate frame coincides with the reference or observer coordinate frame then $X_s = Y_s = Z_s = 0$, and $T = T_0 = (U,V,W)$,which simplifies the equation for optical flow to give :

$$u = \frac{U - xW}{Z} - \alpha xy + \beta(x^2 + 1) - \gamma y \qquad (2.4.1)$$

$$v = \frac{V - yW}{Z} - \alpha(1 + y^2) + \beta xy + \gamma \qquad (2.4.2)$$

The above pair of equations embodies the constraint that the optical flow (u,v) imposes upon the the parameters of rigid motion. Thus all an observer has to do to determine where he is going is to measure the retinal velocity pattern and then use the above pair of equations applied at least five points [1, 13, 17], to determine the 3D velocity of egomotion. Note that there are six velocity components (i.e. three for translation and three for rotation). Unfortunately however, all the six parameters cannot be computed by monocular visual data. This is because of the depth term 'Z' that occurs in the above pair of equations. The depth introduces a scaling effect, whereby other things being equal, multiplying the translational components and the depth by the same constant factor leaves the perceived retinal motion unchanged. Thus for example an object at a certain distance, translating with a certain speed generates the same optical flow field when it is twice as far away and travelling in the same direction with twice the speed.

Thus the monocular observer, lacking depth information, must eliminate the depth factor from the optical flow constraints. This will then imply that the observer's translation can only be determined upto a scale factor. Thus the number of egomotion parameters of interest are five - pertaining to the direction of translation and the rotation.

When the depth variable is eliminated from the above equations we have

$$\frac{x_0 - x}{y_0 - y} = \frac{u + \alpha xy - \beta(x^2 + 1) + \gamma y}{v + \alpha(y^2 + 1) - \beta xy - \gamma x} \qquad (2.5)$$

where $(x_0,y_0) = (\frac{U}{W}, \frac{V}{W})$ represents the direction of translation of the observer's coordinate frame.

The above constraint equation demonstrates the difficulty of motion computation for a monocular observer. It is nonlinear as well of high dimensionality, both this properties in conjunction make the problem difficult ([1, 4, 8, 9, 16, 17]).

It will now be shown that in case the monocular observer can discern a distinguishing feature or mark on the observed surface then the perception problem becomes simpler. Suppose that the surface in view has an easily distinguishable and localized feature at point 'S' whose corresponding image location is (x_s,y_s). In this case we can shift the body origin to the point S and rewrite the optical flow equations as in (2.3). In addition

$$u(x_s,y_s) = u_s = \frac{U_s - x_s W_s}{Z_s}$$
$$v(x_s,y_s) = v_s = \frac{V_s - y_s W_s}{Z_s} \qquad (2.6)$$

Combining equations (2.6) and (2.3) one obtains:

$$u = \frac{u_s + (x_s - x)W_s'}{Z'} + \frac{\alpha xy_s - \beta(1 + xx_s) + \gamma y_s}{Z'} - \alpha xy + \beta(1 + x^2) - \gamma y \qquad (2.7.1)$$

$$v = \frac{v_s + (y_s - y)W_s'}{Z'} + \frac{\alpha(yy_s + 1) - \beta x_s y - \gamma x_s}{Z'} - \alpha(1 + y^2) + \beta xy + \gamma x \qquad (2.7.2)$$

where the 'prime' operator signifies scaling by Z_s, i.e. $W_s' = \frac{W_s'}{Z_s}$. Note that the translational parameters with respect to the observer's frame (i.e. the observers actual translation) are related to the body centered translational parameters by

$$U' = U_s' - \beta + \gamma y_s$$
$$V' = V_s' + \alpha - \gamma x_s \qquad (2.8)$$
$$W' = W_s' - \alpha y_s + \beta x_s$$

The above analysis illustrates the fact that given the ability to estimate the projected velocity of a localized feature accurately, the constraint equations reduce in dimensionality by one.

A similar result may be obtained, as can be expected, when the moving observer is able to track a single feature point so that it appears stationary on the retina at position (0,0). In this case we assume that the tracking motion consists of rotations about the axes that are orthogonal to the line of sight or the optical axis of the lens. The tracking motion is a rotation $(\omega_x,\omega_y,0)$, which is superimposed upon the actual parameters of motion.

Let $S = (0,0,Z_0)$ be the spatial coordinates of the point being tracked. Assume that the observer can track an environmental point and hold it steady on the optical axis (Z axis). Therefore the optical flow field will have a singularity at the origin of the retinal frame, where the flow value is zero. At the time of observation, the tracked point tends to move along the observer's optical axis (figure II).

Consider an observer moving with translation (U,V,W) and rotation (α,β,γ). Then, if the body frame origin is taken to be at S, from equation (2.8), remembering that $U_s = V_s = 0$:

$$U' = \frac{U}{Z_0} = -B$$
$$V' = \frac{V}{Z_0} = A \qquad (2.9)$$
$$W_s = W$$

furthermore the optical flow equation (2.3) becomes:

$$u = \frac{-xW}{Z} - Axy + B[1 - \frac{Z_0}{Z} + x^2] - \gamma y$$
$$v = \frac{-yW}{Z} - A[1 - \frac{Z_0}{Z} + y^2] + Bxy + \gamma x \qquad (2.10)$$

where $A = \alpha + \omega_x$ and $B = \beta + \omega_y$.

Eliminating Z from the above we have:

$$\frac{u + Axy - B(x^2 + 1) + \gamma y}{v + A(1 + y^2) - Bxy - \gamma x} = -\frac{B + xW'}{A - yW'} \qquad (2.11)$$

where $W' = \frac{W}{Z_0}$.

The constraint equations derived above are similar in form to equation (2.5). However, in this case the dimensionality of the parameter space has been reduced from five to four *without increase in the degree of nonlinearity of the constraint.* It is important to note that the observer can determine his direction of translation since from equation (2.9) we have

$$U' = \frac{U}{Z_0} = B$$
$$V' = \frac{V}{Z_0} = -A$$

Thus even without explicitly measuring his tracking motion, the observer can determine the scaled translation (U',V',W'). We next examine the constraint equation (2.10) and show how it may be used to actively compute the direction of translation.

3. Stereo tracking

It can be expected that stereoscoping viewing can simplify the task of motion perception. Binocular imaging system does introduce a new complication in that in addition to the task of retinal motion estimation, one must also accomplish stereo fusion. However stereo fusion is a simpler task than optical flow estimation, and a recently published algorithm is reportedly able to handle this task reasonably satisfactorily [14]. The advantages of stereo imaging for analysing motion are:

1. The motion constraint equation is linearized.

2. The epipolar constraint is a powerful aid in handling the "correspondence problem" for both stereo fusion as well as retinal motion estimation. (In this section the reason for this will be sketched, but it will not be detailed)

The monocular imaging geometry described previously is augmented by two other coordinate frames located at the points $(d,0,0)$ (the left eye frame) and $(-d,0,0)$ (the right eye frame) respectively. The central frame is an imaginary "head frame" and the two other frames are the camera or "eye" frames. The situation is depicted in (figure III). In this scheme there is no vergence between the two eye frames (rather the eyes verge at infinity). This means that the corresponding axes of all the coordinate frames are parallel. Furthermore, it is assumed that the frames are rigidly attached to each other.

The tracking action is with respect to the head frame. Now if the head frame is tracking a feature point S_h at $(0,0,\rho)$ then its image on the left and right eye frames are at $(-e,0)$ and $(e,0)$ respectively. The relation between the depth ρ and e is

$$\rho = \frac{2d}{2e} = \frac{d}{e}$$

Once again for simplicity of explanation, consider the relative motion between the observer's head frame and the observed rigid scene, as due to egomotion. The motion parameters are the translational velocity (U,V,W) and the rotational velocity (α,β,γ). The observer's tracking movement is confined, as before, to the rotation $(\omega_x,\omega_y,0)$, with respect to the head frame. The tracking motions executed by the the eye frames include this rotation *plus* translations in depth of $-d\omega_y$ and $d\omega_y$ respectively.

Consider an image location (x_l,y) in the left frame, and its stereo pair (x_r,y) in the right frame. The disparity is given by

$$\delta = x_r - x_l = \frac{2d}{Z}$$

where Z is the depth of the point in space giving rise to the stereo pair.

The motion parameters are as before (U,V,W) and (α,β,γ), with respect to a hypothetical head frame located between the two stereo coordinate frames. The head frame is assumed to track the environmental feature S_h (the subscript refers to the fact that the nomenclature is with respect to the head frame). Therefore equations (2.9) hold. The motion parameters with respect to the stereo frames are:

$$L: \ T_l = T_l^b + T_l^{tr} \qquad \Omega_l = \Omega_l^b + \Omega_l^{tr}$$
$$R: \ T_r = T_r^b + T_r^{tr} \qquad \Omega_r = \Omega_r^b + \Omega_r^{tr}$$

where the subscripts l or r refer to the left or right frames, and the superscripts b or tr refer to body parameters (or representating actual motion) and motion induced due to the tracking motion respectively. These components can be expanded to

$$T_l^b = (U, V + \gamma d, W - \beta d) \qquad \Omega_l^b = (\alpha, \beta, \gamma)$$
$$T_r^b = (U, V - \gamma d, W + \beta d) \qquad \Omega_r^b = (\alpha, \beta, \gamma)$$

and

$$T_l^{tr} = (0, 0, -\omega_y d) \qquad \Omega_l^{tr} = (\omega_x, \omega_y, 0)$$
$$T_r^{tr} = (0, 0, \omega_y d) \qquad \Omega_r^{tr} = (\omega_x, \omega_y, 0)$$

It can be seen that the motion of the tracked point, is given as $T_s = (0, 0, W)$ in both the left and right frames. The rotation of these frames is also the same, namely (A, B, γ). Finally, the tracked point is located with respect to the two frames as: $S_l = (-d, 0, \rho)$ and $S_r = (d, 0, \rho)$. Therefore from equation (2.3) we get the optical flow constraints for the left eye as:

$$u_l = \frac{-x_l W}{Z} - Ax_l y + B\left(1 - \frac{\rho}{Z} + x_l^2 + \frac{x_l d}{Z}\right) - \gamma y$$
$$v_l = -\frac{yW}{Z} - A\left(1 - \frac{\rho}{Z} + y^2\right) + B\left(x_l y + y\frac{d}{Z}\right) + \gamma\left(x_l + \frac{d}{Z}\right)$$

where $A = \alpha + \omega_x$ and $B = \beta + \omega_y$. In the above equation, making the substitution $\left(Z = \frac{2d}{x_r - x_l}\right)$ we have:

$$u_l = x_l\varphi + B\left(1 - \frac{\rho}{Z}\right) - \gamma y$$
$$v_l = y\varphi - A\left(1 - \frac{\rho}{Z}\right) + \gamma\frac{(x_l + x_r)}{2} \qquad (3.1)$$

where $\varphi = -\frac{W}{Z} - Ay + B\frac{x_l + x_r}{2}$. similarly the optical flow for the right eye is given by:

$$u_r = x_r\varphi + B\left(1 - \frac{\rho}{Z}\right) - \gamma y$$
$$v_r = y\varphi - A\left(1 - \frac{\rho}{Z}\right) + \gamma\frac{(x_l + x_r)}{2} \qquad (3.2)$$

From the above equation we get:

$$\varphi = \frac{u_r - u_l}{x_r - x_l}$$

which leads to a constraint equation in two parameters:

$$v_r = v_l = y\frac{u_r - u_l}{x_r - x_l} - A\left(1 - \frac{\rho}{Z}\right) + \gamma\frac{x_l + x_r}{2} \qquad (3.3)$$

This with stereo tracking it is possible to obtain a linear constraint in two unknowns at every point of measurement.

4. Experiments

We have carried out some preliminary experiments on artificial images to date. Although the above analysis assumes the velocity representation, the experiments used data generated by small discrete displacements. We also assume that optical flow is known at each point. Using binocular vision and tracking A and γ are

computed, from which we can recover the other parameters.

The algorithm used to recover A and γ is as follows:

(a) Obtain possible stereo correspondences by epipolar constraints, i.e. the difference in the y values in the two camera's image frame has to lie within a certain value which we shall call the radius.

(b) Calculate the depth of the point by the correspondence.

(c) Throw away all correspondences which give extreme values of (depth of point - ρ) where ρ = depth of tracked point.

(d) Repeat step c until the number of points has been reduced to some threshold (typically the original number of points).

(e) Calculate the coefficients of A and γ in equation(3.3) for the remaining points, and apply the least squares method to obtain A and γ.

The experiments were performed for values of F(focal length) ranging from 35mm to 200mm, d(stereo baseline/2) ranging from 4 cm to 20 cm, θ (angle of rotation) varying between 2 degrees and 5 degrees and additive noise of upto 20 percent. We found that the algorithm was quite stable within these limits, recovering A and γ to within 10 - 25 percent accuracy. As the radius(distance between epipolar lines for correspondence) increased the error increased. Further, if steps (c) and (d) of the algorithm were not carried out, the errors were found to be much bigger, specially as the radius became large. The results are summarized in the following table:

radius	av. false match count	γ	A
0	0.76	0.0640	0.0647
1	2.48	0.0634	0.0539
2	3.83	0.0638	0.0533
3	5.27	0.0636	0.0526
5	7.83	0.0632	0.0510
10	15.26	0.0645	0.0485

The parameters relevant to the above table are (the unit of length is one pixel width):
Focal length = 1000
stereo baseline = d = 1000
Rotation = (A,B,γ) = (0.0688 , 0.0229 , 0.0688)
Translation = (U , V , W) = (- 227 , V = 611 , W = 34)
Percentage of noise = 10

The algorithm works with large amounts of additive noise because most of the noise points get removed in step (c) of the algorithm. Other points whose depth is calculated to be very different from ρ also get removed, leaving points for which the A coefficient in equation (3.3) are quite similar which gives better results with least squares. The error is due to two factors:

(a) Discretization: This becomes specially important when the optical flow or the depths are small.

(b) Wrong correspondences: We hope to reduce the number of wrong correspondences by using some better statistical techniques and integrating these with hough transform.

5. Conclusion and summary of work in progress

A new direction in the study of the perception of rigid motion from visual cues, has been explored. Some of the problems inherent in the rigid motion problem and their possible solutions were examined. The idea that tracking environmental points may be beneficial to navigation has previously been put forward by Cutting [3]. However, his analysis is largely qualitative. We have developed a mathematical framework for the active navigation employing tracking. Binocular motion perception has been considered previously by [7, 15] and more recently by [19]. The results reported here show that tracking a prominent environmental feature point is advantageous in both monocular and binocular imaging situations.

In summary the following are the main results reported here:

1. Monocular tracking reduces the dimensionality of the parameter space by one without increasing the complexity of the constraint equation.
2. Employing binocular tracking reduces the constraint to linear equations in two unknowns.
3. The tracking motion need not be known to determine the observers translation and rotation about the Z axis.

We are currently working on the mathematical formulation of the problem for a verging binocular imaging system. A concurrent project by the Robotics Group at the University of Rochester involves the design and fabrication of the binocular tracking mechanism using two video cameras and a computer controlled mount. This will allow us to test the constraint equations on real images.

6. Acknowledgements

The preparation of this paper has been greatly aided by the critical comments of Chris Brown and Dana Ballard. We also thank Yannis Aloimonos and Paul Chou for many helpful suggestions. The assessment of the reviewers of this paper helped to improve its contents. This work was supported by a National Science Foundation grant number DCR-8405720.

7. References

1. A. Bandopadhay, Constraints on the Computation of Rigid Motion Parameters from Retinal Motion., Tech. Rep. 168,

Department of Computer Science, The University of Rochester., October 1985.

2. S. T. Barnard and W. B. Thompson, Disparity Analysis of Images, *IEEE Trans. PAMI 2*, (1980), 333 - 340.

3. J. E. Cutting, Motion Parallax and Visual Flow: How to Determine Direction of Locomotion. (Paper presented at the 4th meeting of the International Society for Ecological Psychology, Hartford, CT.,1982), Dept. of Psychology, Cornell University, 1982.

4. J. Q. Fang and T. S. Huang, Solving three dimensional small-rotation motion equations: Uniqueness, algorithms, and numerical results, *CVGIP 26*, (1984), 183-206.

5. E. C. Hildreth, Computations Underlying the Measurement of Visual Motion, *Artificial Intelligence 23*, (1984), 309-354.

6. B. K. P. Horn and B. G. Schunck., Determining Optical Flow, *Artificial Intelligence 17*, (1981), 185-204.

7. Jenkin, The stereopsis of time-varying images, Tech. Rep. RCBV-Tech. Rep.-84-3, University of Toronto, Comp. Sci Dept., 1984.

8. H. C. Longuet-Higgins and K. Prazdny, The interpretation of a moving retinal image, *Proc. Royal Soc. London B 208*, (1980), 385 - 397.

9. H. C. Longuet-Higgins, A Computer Algorithm for Reconstructing a Scene from Two Projections, *Nature 293*, (September 1981), 133-135.

10. H. H. Nagel and B. Neumann, On 3-D Reconstruction from Two Perspective Views, *IJCAI*, , 1981, 661-663.

11. H. H. Nagel, Overview on Image Sequence Analysis, in *Image Sequence Processing & Dynamic Scene Analysis*, T. S. Huang (ed.), 1983.

12. H. Nagel, Displacement Vectors Derived from Second order Intensity Variations in Image Sequences, *CVGIP 21*, (1983), 85 - 117.

13. K. Prazdny, Egomotion and Relative Depth Map from Optical Flow, *Biol. Cybernetics 36*, (1980), 87-102.

14. K. Prazdny, Detection of Binocular Disparities, *Biol. Cybern. 52*, (1985), 93-99.

15. W. Richards, Structure from stereo and motion, *J. Opt. Soc. Am. A 2 (2)*, (February 1985), 343-349.

16. J. H. Rieger and D. T. Lawton, Determining the Instantaneous Axis of Translation from Optic Flow Generated by Arbitrary Sensor Motion, *COINS tech. report 83-1.*, , January 1983.

17. R. Y. Tsai and T. S. Huang, Uniqueness and Estimation of Three Dimensional Motion Parameters of Rigid Objects with Curved Surfaces, *IEEE. Trans. PAMI 6*, (January 1984), 13-27.

18. S. Ullman and E. Hildreth, The Measurement of Visual Motion, in *Physical and Biological Processing of Images (Proc. Int. Symp. Rank Prize Funds. London)*, O. J. Braddick and A. C. Sleigh (ed.), Springer-Verlag, September 1982, 154 - 176.

19. A. M. Waxman and J. H. Duncan, Binocular Image Flows: Steps toward Stereo - Motio Fusion, CAR-Tech. Rep.-119, Computer Vision Laboratory, University of Maryland., May 1985.

Imaging geometry and basis for body centered motion representation

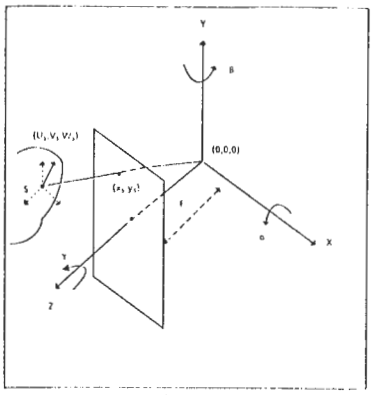


Figure I

Tracking Motion showing apparent translation in depth of tracked point

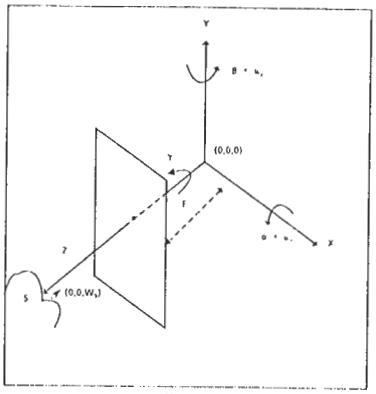


Figure II

Tracking in Stereo

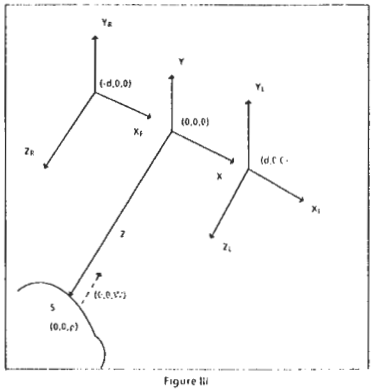


Figure III

COMBINING VISUAL AND TACTILE PERCEPTION FOR ROBOTICS

by
J.C. Rodger † and R.A. Browse † ‡

†Department of Computing and Information Science
‡Department of Psychology
Queen's University
Kingston, Ontario

ABSTRACT

This paper describes a system which integrates visual and tactile information for robotic perception. Objects selected from a predefined set are recognized and located by using knowledge of visual and tactile imaging conditions along with detailed object models. In each of the perceptual modalities, simple image features are extracted and used to invoke object possibilities. In a combined interpretation process, consistency is enforced among possibilities. The complementary nature of visual and tactile constraints results in determination of object identity and location in situations for which very few features are extracted.

Ce papier decrit un système qui combine de l'information visuel at tactile afin d'accomplir la perception robotique. Les objects, selectionnés d'un ensemble prédéfiné, sont reconnus et leur positions sont determinées grâce à la connaissance des conditions visuelles et tactiles de l'image ainsi que des modèles. Pour chaque capacité perceptuelle, des traits simples sont extraits et utilisés pour determiner des cas possible d'object. En combinant les interprétations, la consistence est enforcée parmis les possibilités. La nature complémentaire des restrictions visuelles et tactiles resulte en l'identification d'un object et sa position dans les cas ou peu de traits sont extraits.

1. Introduction

There has been considerable effort in recent years in applying computer vision and artificial intelligence approaches to the problems of robotic perception (Casasent & Hall, 1984; Pugh, 1984; Rosenfeld, 1984). Robots equipped with various types of sensors have been proposed to address the challenge of increasing automation, while retaining the flexibility and reconfigurability that otherwise requires the presence of a human operator.

One way for robotic perception systems to meet this goal is to integrate the input from two or more sensor modalities in order to take advantage of their complementary characteristics (Browse & Lederman, 1985a). The task demands of a varying environment may include recognition and manipulation of relevant objects. This suggests the utility of integrating visual sensing with tactile/kinesthetic sensing. Recent progress in tactile sensor technology (Dario & De Rossi, 1985), tactile feature extraction (Kinoshita, 1983; Ellis, 1984) and the development of tactile sensing system specifications (Harmon, 1984; Grimson & Lozano-Perez, 1984; Browse & Lederman, 1985b) all support the inclusion of machine touch in multisensory robotic perception systems.

Another reason for interest in such a system is its potential for modeling processes related to human perception studies. There is now empirical evidence for the ability of adult subjects to rapidly and accurately identify familiar objects using only the haptic sense (Klatzky, Lederman & Metzger, 1985). This recent research clarifies earlier results which suggested that tactile performance was poor for tasks in which the stimuli were two-dimensional random shapes. These recent results suggest that tactile perception has access to complex object models and it may be realistic to assign more than just a supplementary role to the tactile/haptic mode.

Other studies have shown that in the classification of three dimensional objects, adult subjects tend to use common dimensions when using either vision alone or touch alone (Garbin & Bernstein, 1984). Size and shape were used in both modalities. This study suggested that object attributes were perceived similarly and that the relationships among attributes were treated much the same in vision and in touch.

There is further evidence for common underlying processes/representations in the developmental perceptual literature. Rose, Gottfried and Bridger (1983) found that infants showed cross-modal transfer in a paired recognition paradigm. Following haptic experience with objects they demonstrated recognition of the objects when they were presented visually.

In summary, the psychological literature demonstrates the existence of powerful tactile-based recognition abilities, plus common representational and processing factors. Implementing a system which demonstrates these abilities may lead to improved understanding of the nature of such processes and representations. Their existence suggests the utility of approaching the problems of robotic perception with the human system as a model.

2. Intersensory Robotic Systems

Luo, Tsai, and Lin (1984) have described a system which uses input from an overhead camera along with a pair of tactile sensors mounted on the jaws of a robot gripper. The visual component of the system extracts a series of moment-based features from a binary image of the object. The vector of features obtained may uniquely identify the object presented. If a unique identification is not possible based on the visual input, then the system closes the gripper jaws on the object in order to obtain two tactile images. The orientation of the gripper is selected on the basis of the object possibilities which remain to be disambiguated plus a decision tree constructed during a training phase. The arcs of the tree give the approach angles for discriminating among the possibilities at that level. Moment-based features are extracted from the tactile images obtained in this way and matched against the feature vectors for the

remaining candidate objects. Additional tactile image pairs may be acquired by this process if object identity is still uncertain. If a leaf node of the decision tree is reached which contains more than one object possibility, the system may move the object by rotating it to show another stable orientation, and repeat the identification process.

Another approach has been described by Allen and Bajcsy (1985). Their system uses a stereo camera pair plus a tactile sensor modeled after a finger, mounted on the end of a robot arm. The goal is to recognize and locate single, rigid objects by computing three dimensional features and matching these to objects in a database of models. The models are hierarchically structured, with an object node at the root, and entities described as components, holes or cavities at the first level. Components are topologically and/or functionally distinct parts which are described in terms of constituent surface patches. The first level nodes contain both attributes of individual parts and relations among them. The recognition process has an initial input-driven stage, followed by model invocation and a model-driven stage. The system uses the stereo vision subsystem to derive a sparse set of 3-D points on the object. Then the tactile subsystem is guided by this representation in further exploration of the object. Surfaces are traced and holes and cavities measured. As object primitives become available, they are used for matching to parts of models. Both model and relational consistency can be required in these matches. If only one candidate object remains at this stage, the system uses the model to guide additional verification processing. If multiple possibilities remain, a weighted probability measure is used to select one to use as a working hypothesis for further model-guided exploration. The local coordinate systems of holes and cavities, plus the normals to surface patches are used in computing a transformation on the object to locate it in the workspace. Verification proceeds by using the computed transformation and selected model to determine additional component, hole and cavity elements to explore with the finger sensor. The process stops when identification to within some confidence measure is achieved.

The preceding approaches to the problem of integrating two sensor modalities for robotic perception illustrate quite distinct traditions. The former uses a recognition strategy based on 2-D pattern classification techniques to decide the identity at each stage. The latter adopts the common computer vision strategy (Marr, 1982) of developing 3-D primitives for matching to models. The difficult problem of context-free derivation of 3-D primitives from 2-D images is divided into sub-problems by only requiring a rough 3-D computation from the vision module and relying on the tactile modality to fill in the details.

In our approach to the problem, we recognize the essential 3-D nature of the task, but attempt to combine the simplicity of 2-D features, with the inherent 3-D advantage of the haptic system. We believe that an alternative formulation of the computer vision problem may be applicable to the general recognition problem. The possibility that simple image primitives may invoke relatively high level explanations has been explored by several authors (Browse, 1982; Witkin & Tenenbaum, 1983; Lowe, 1984; Biederman, 1985). The current system attempts to test the hypothesis that simple features may be used to access object models early in the processing cycle. This facilitates the early application of object knowledge to the recognition problem.

For the vision component we chose the simplest practical image features - line segments. The tactile component also uses elementary features such as indications of edges and region boundaries.

Our approach also differs from those described above in its assumption of roughly equivalent roles for the two component systems. This contrasts dramatically with that of Luo et al., who assign a secondary role to the tactile modality, invoking it only to disambiguate visually indistinguishable objects. This also distinguishes our approach from that of the Allen & Bajcsy, whose system has a rigid ordering on the use of the perceptual modalities.

3. System Description

The system outlined below is implemented in Zetalisp and runs on a Symbolics LM3600. The tactile component was developed in Franz Lisp on a VAX 11-780, and is described in more detail in Browse (1986).

The system can be broadly classified as a feature-based object recognition system, using features simulated as if they were obtained from two distinct sensor modalities. The system takes input features derived from one of a set of object models in its database and attempts to determine which of the objects gave rise to the input. In addition, the present system extracts the three-space location and orientation of the identified object. These are not distinct subtasks, but are computed together as a single goal. Figure I provides a diagrammatic overview of the entire system.

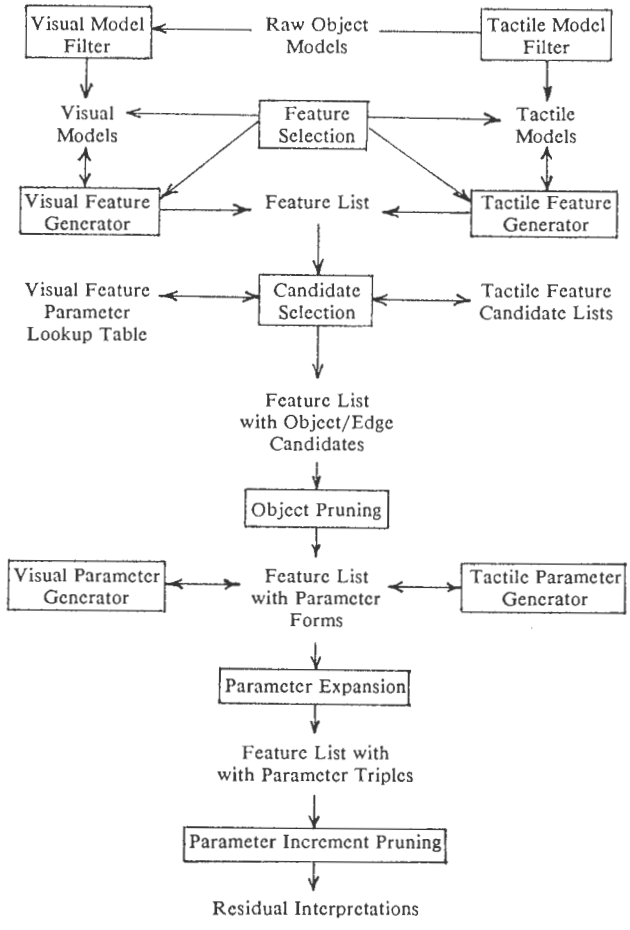


Figure I. System overview diagram for the Visual-Tactile Perceptual system.

An interactive shell permits testing of the system under varying conditions of assumed input and viewing parameters. The user chooses one of several pre-determined sets of viewing parameters, an object from the model database and a three parameter transformation on that object to locate and orient it in the workspace.

The specified object and transformation are stored for subsequent access by the display functions and input feature generators. Next, the user selects input features to simulate. Any number of features and any mixture of features from the tactile and visual domains may be selected. In the current implementation the system uses whatever features appear as input to constrain the possibilities for the identity and location of the input object as much as possible. It is assumed that only a single object from among the known models is present.

3.1. World Assumptions All the objects that the system knows about, including the "camera" that generates simulated visual images, and the "touch-pad" that produces simulated tactile input are defined in a three-space coordinate system, subsequently referred to as the world coordinate system (WCS). For purposes of developing the system, the convention was adopted that the units of the WCS were centimeters. This convention gives an absolute reference for the specification and comparison of object sizes, camera parameters, and touch-pad dimensions.

The assumed environment is simplified by adopting the additional convention that the XY plane represents the "floor" of a workspace. We assume that a distinct object corresponds to a gravitationally stable orientation, and rests on this floor. The result is that the system describing an object's position and orientation in the WCS has three free parameters - two translational and one rotational. Figure II shows an object in the simulated workspace with the tactile sensor in contact.

3.2. Object Models The system has a small set of object models (9 distinct objects in test versions). Each has a home position, centered on the WCS origin. The descriptions are edge-based - wire frame models of polyhedra. Each model is implemented as a lisp object, an n-ary tree of depth three. The root is an identifier for the object. Branches at the first level are object faces. Each face is a subtree, with an identifier at the root and a subtree for each edge. Edges are lists consisting of an edge identifier, a number indicating the curvature of the edge, plus two lists which are the homogeneous coordinates of the end points of the edge in the WCS. The tactile component includes features based on the surface curvature, and uses nonzero curvatures when run independently, but the current version of the visual component operates only with straight edges.

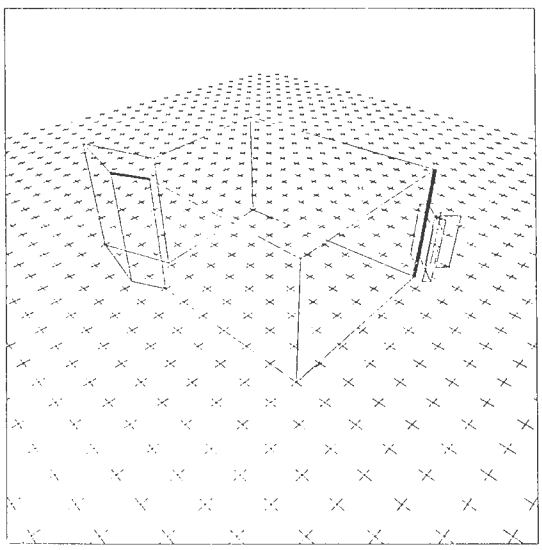


Figure II. Simulated robotic workspace.

While there is a single set of external models, each subsystem builds its own internal representation. For the tactile component, the third dimension is orthographically projected out of the models by assuming that its objects have constant cross-section, with each edge either parallel or orthogonal to the XY plane of the WCS. To facilitate this simplification, the shared model descriptions place the top face of the object in the first position. The tactile component can ignore the rest of the description. In building the internal representation, the subsystem computes additional properties from the raw model. These include the length of each edge, its slope in the XY plane, and the angle between successive edges as the outer border of the object is traced. The result of the model filtering routines is a list of object identifiers, each with a "surfacelist" property, the value of which is a list of surfaces, each in turn a list of the features associated with a surface - curvature, length, and so on.

The filter for converting the raw model descriptions to the internal representation used by the visual component is simpler. The internal model ignores the face level of the raw model, which is included in the current version in anticipation of the future use of face-based features, and the inclusion of hidden line/surface algorithms in display and other stages. The result of the visual filtering of the raw models is a table of end point coordinates indexed by object and edge identifiers.

3.3. Features For vision, only straight line segments are currently implemented as features. The user selects one edge of the chosen object. The feature is the perspective projection of the selected edge of the current object. The actual information stored consists of the image coordinates of the end points of the resulting line segment.

The choice of tactile features is based on a study which suggests that certain features form separable dimensions in human touch (Lederman & Browse, 1986). The available tactile features are a subset of those which are extracted by Ulug (1986) using a Barry Wright Corp. Sensofelex tactile sensing system. There are three categories of simulated tactile features - corners, edges, and flush contacts between the touch-pad and an object surface. Given the object, transformation, edge, and feature type, the system computes the location and orientation of the touch-pad that would generate the specified feature. The actual information stored is this pad transformation plus the feature type. From this point in the processing, only the feature information is used in arriving at an interpretation.

3.4. Model Possibilities As features are entered the system uses them to get lists of object/edge possibilities for each one. For both domains, this is accomplished by accessing pre-computed data structures. These store all possible object/edge combinations for each feature category. In the visual component, since only straight line segments are currently used as features, it is necessary to categorize these in a way that allows some discrimination among all the object model edges. They are parameterized by the computed length and angle of intersection with the X axis of the corresponding image line. For a single viewpoint, limited field of view, and only three free orientation and translation parameters a given object edge will only image to some subset of all the possible combinations of length and angle.

As a result, this parameterization provides some initial screening in determining object possibilities for a given image feature.

For the tactile component, both the shape of the object and the size of the touch-pad place some restrictions on the possibilities that are consistent with a particular input feature. These derive partially from the restriction on edge and corner features at the junction of two surfaces which form a concave angle. The relative size of the pad and a surface may also exclude some surfaces from the list of possibilities for a flush feature.

3.5. Precompiling Assembling lists of possibilities for visual features is accomplished by exhaustively varying the values of each of the three free parameters over selected ranges. The image parameters for each resulting line are computed and the object/edge combination is added to a table location indexed by the parameter values. Because of the computational demands of this operation, it is performed offline, and the resulting tabular data structure is loaded into the system as required for a given session.

For the tactile domain, the reduced number of edge entries for each object in the internal representation, plus the absence of perspective ambiguity make it is feasible to assemble the lists of possibilities for each feature category on-line. This occurs once, at the beginning of a session, after the object models have been loaded. The identifier for each of the feature categories has as a property the list of object/edge possibilities which could give rise to it.

3.6. Object Consistency Once a list of features, each with an associated list of object/edge possibilities has been set up the next phase of the recognition process can begin. This involves pruning inconsistent possibilities in two stages. Since one of the assumptions of the system is that only a single object is present, an initial pass over the list of features removes from each list of candidates any which are not supported by consistent (ie. the same object) candidates for each of the other features. The power of this stage depends on the degree to which different features invoke different subsets of object possibilities as well as the variation among objects.

3.7. Constraint Addition The next operation is the addition to the simple possibilities of constraints on the three free parameters. The constraints derive from the feature acquisition conditions and the nature of the features themselves. The limits on the parameters are be expressed in different ways, depending on the type of the associated feature.

For the visual features, the parameter constraints are expressed either as a triplet of discrete values or as a range of values for the rotational parameter and expressions in this parameter for each of the two translational parameters. Each object/edge possibility for an input feature is used to retrieve the corresponding WCS endpoints. The Z coordinates can be used with the inverse perspective transformation to locate the world points corresponding to the image coordinates of the line segment feature, under the assumption that the particular edge gave rise to it.

For object edges which are not parallel to the axis of rotation, correspondence between two pairs of point coordinates allows the computing of discrete values for each of the three parameters. These would take the selected edge into the obtained line segment under the known viewing conditions. Where the assumed correspondence yields world coordinates which are inconsistent with any three parameter transform of the edge in question, the edge is dropped from the list of possibilities.

For object edges which are parallel to the axis of rotation, correspondence between two pairs of points fixes the WCS location of the edge, but leaves the object free to rotate about the edge. The resulting constraint form is given as a range of rotation values plus expressions in the angle of rotation for each of the translation parameters.

For the tactile component the form of the parameter constraint for each possibility depends on the feature type. Edge features occur when the touch-pad is in flush contact with an object surface so that the boundary of the surface falls within the borders of the pad. This allows computing of discrete values for each of the three parameters. Flush contact with a surface, without a boundary contour fixes only the angle parameter. The system uses the hypothesized object surface dimension and the angle to determine a range for one translational parameter, and then the other can be expressed in terms of the angle and first translation. Finally, contacting only the boundary between two surfaces generates a corner feature. This is analogous to the visual case of a candidate edge parallel to the axis of rotation. The location of the feature acquisition is fixed; however, the object may rotate about the corner. The constraint form expresses the rotation parameter as a range, and the translation parameters as expressions in the angle of rotation. Figure III demonstrates an object/contact possibility for a tactile feature in which the object has been plotted in increments through the range of allowable transformations.

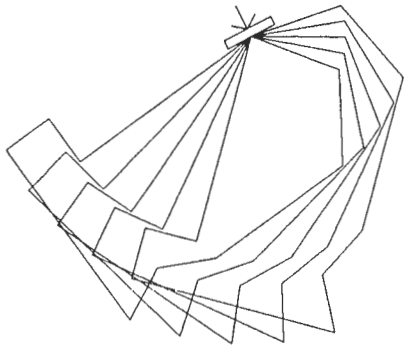


Figure III. An object/contact possibility for a tactile feature.

3.8. Constraint Expansion and Increment Pruning The parameter constraint forms are added to surviving object/edge possibilities for each feature. Then the system expands these constraint forms by subdividing ranges into equally spaced increments and evaluating expressions in the variables which were previously described only as ranges. The result for each possibility is a list of triples of values for the three parameters. Once the constraints are expressed in this form, comparisons across features can be made, and inconsistent interpretations can be dropped.

The system does an exhaustive filtering operation in order to determine which interpretation possibilities, if any, survive the increment pruning stage. One feature is chosen and its interpretations are checked against those of all the remaining features. The system iterates over the parameter triples for each possibility, and searches for compatible possibilities and increment triples associated with each of the other features. A

candidate possibility is eliminated if the search fails to find a compatible possibility for the same object for every other feature. Compatibility is defined as the matching of parameter values to within specified tolerances. Within a retained possibility, parameter triples are dropped if they are not supported by compatible triples for a consistent possibility for every other feature.

This pruning process yields a reduced list of possibilities with reduced lists of associated parameter triples These are interpretations which have support across all features, and should be empty only if the input features were not generated from one of the object models in the system's database. The list should be unchanged only in the case of a single input feature. Otherwise, the reduction depends on the type of input features, as well as the tolerances which define compatibility. In practice, if three or more features are present the possibilities generally reduce to only a few parameter combinations for the single correct object. This is especially true if the tightly constraining features types predominate. For the visual component these are edges which are not parallel to the WCS Z-axis. For the tactile component these are edge contacts and, to a lesser degree, flush contacts.

4. Examples

This section presents examples of the operation of the system described above. Figures IV and V each illustrates a configuration in which two features, one from each modality, are input to the system. The visual features and the specified placement transform are the same in both examples. The parameter values are 15 degrees of rotation and five units of translation in each of the X and Y dimensions.

In Figure IV, the tactile feature is a corner. These two features allow the system to prune the initial possibilities from 84 for the visual feature and 52 for the tactile feature to a single compatible parameter triple for the correct object. The obtained placement transform parameter values are 16.4 degrees, plus 5.2 units in each of X and Y. Figure V shows the same configuration, but with a flush contact for the tactile feature. Initial possibilities numbered 84 and 49 for the visual and tactile features respectively. These were pruned to four parameter triples for the correct object. One of these is illustrated in Figure VI. The surviving triples consist of increments "sliding" the object along the flush contact. These triples bracket the specified placement transform values.

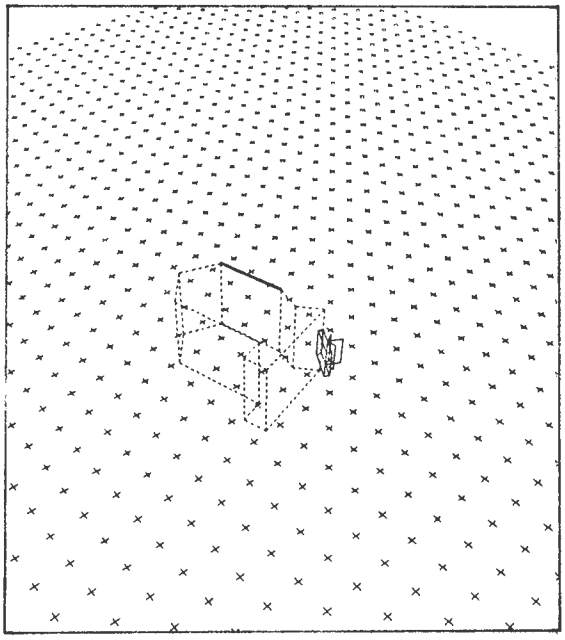


Figure IV. Object with one visual feature and tactile corner feature.

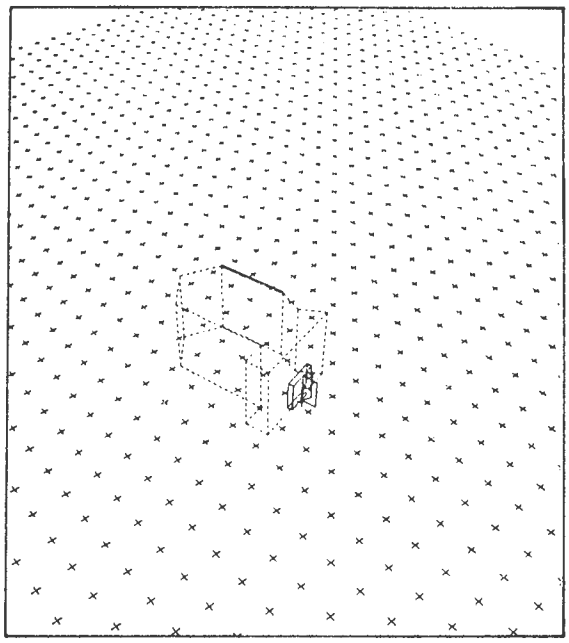


Figure V. Object with one visual feature and tactile flush feature.

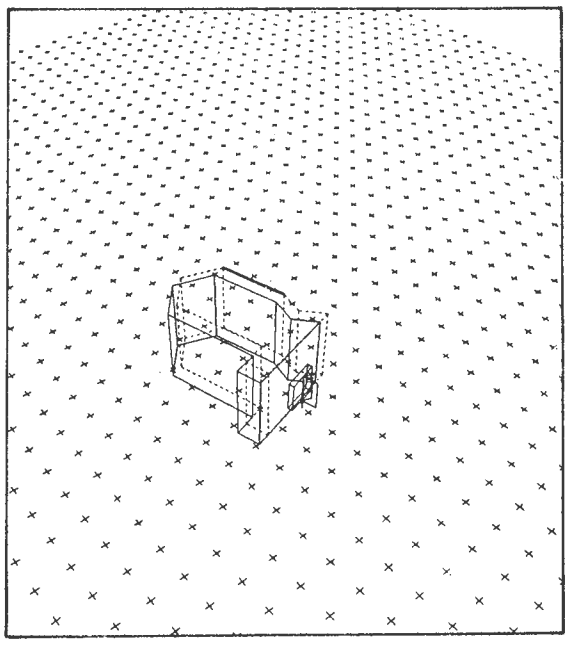


Figure VI. Specified placement (dashed lines) and compatible parameter placement(solid lines).

5. Conclusions

This paper has outlined the simulation of a feature-based system for integrating visual and tactile input for robotic object recognition. Tests of the system confirm the feasibility of using simple visual and tactile features for early invocation of object models in order to eliminate most of the possible interpretations for a single solid object. While one feature has little selective power, two or three features frequently prune the interpretation possibilities down to only a few parameter triples for the correct object.

Of course, the object set is rather small - only 9 distinct objects in tests so far - and since the features are simulated the input data is noise free. However, the techniques described here will soon be applied to a real robotic setting.

One of the problems of the current approach revealed by testing is the potential for explosive computational complexity under worst case conditions. For our system this occurs when the input features allow a range of the angle parameter for each possibility under each feature. In this case the system must expand each parameter constraint over many increments and then exhaustively check them for compatibility. We propose to avoid this difficulty by selecting features intelligently. The most straightforward way to accomplish this is to impose the consistency requirements over a subset of features before admitting less selective ones into the computation. Techniques to take into account the requirements for disambiguation within the current set of object possibilities are also possible (see Browse, 1983). In addition, we plan to revise the parameter expansion and consistency checking algorithms to operate incrementally, so that expansions are computed to greater accuracy only as consistent interpretations at previous levels require it.

While the use of line segment features in this system has demonstrated the feasibility of the approach, we realize that a richer feature is desirable. The sources cited above in relation to employing simple 2-D features offer suggestions such as parallelism, co-termination, symmetry etc. Subsequent versions of the system should incorporate additional visual features as well as the improved possibility pruning strategies outlined above.

Acknowledgments

The authors wish to thank Mike Jenkins for the use of the Symbolics LM3600 on which the system was implemented. This work was supported in part by the National Science and Engineering Research Council under Operating Grant number A2427.

6. References

Allen, P., & Bajcsy, R. Object recognition using vision and touch. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (Vol. 2)*, Los Angeles CA., August 18-23, 1985, 1131-1137.

Biederman, I. Human image understanding: Recent research and a theory. *Computer Vision, Graphics, and Image Processing*, 1985, *32*, 29-73.

Browse, R.A. Knowledge-Based Visual Interpretation Using Declarative Schemata. Ph.D. Thesis, Department of Computer Science, University of British Columbia. Technical Report 82-12, 1982.

Browse, R.A. Computational Selection of Processing Location in Vision. *Proc. Fifth Annual Conference of the Cognitive Science Society*, Rochester, New York, 1983.

Browse, R.A. Feature-Based Tactile Object Recognition. submitted December 1985 to *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Browse, R. A., & Lederman, S. J. A framework for robotic perception. *TR-85-165*, Department of Computing and Information Science, Queen's University, Kingston, Ontario, Canada, 1985.

Browse, R.A. & Lederman, S.J. Feature Based Robotic Tactile Perception. *COMPINT-85* IEEE Conference on Computer Aided Technologies, Montreal, Sept 1985, pp. 455-458.

Casasent, D. P., & Hall, E. L. (Eds.). *SPIE Proceedings: Intelligent Robots and Computer Vision. 521*, 1984.

Dario, P., & De Rossi, D. Tactile sensors and the gripping challenge. *IEEE Spectrum*, 1985, *22*(8), 46-52.

Ellis, R. E. Extraction of tactile features by passive and active sensing. In D. P. Casasent, & E. L. Hall, (Eds.), *SPIE Proceedings: Intelligent Robots and Computer Vision. 521*, 1984, 289-295.

Garbin, C. P., & Bernstein, I. H. Visual and haptic perception of three-dimensional solid forms. *Perception & Psychophysics*, 1984, *36*, 104-110.

Grimson, W. E. L., & Lozano-Pérez, T. Model-based recognition and localization from tactile data. *Proceedings: IEEE International Conference on Robotics*, Atlanta GA., March 13-15, 1984, 248-255.

Harmon, L. D. Automated touch sensing: A brief perspective and several new approaches. *Proceedings: IEEE International Conference on Robotics*, Atlanta GA., March 13-15, 1984, 326-331.

Kinoshita, G. Representation and tactile sensing of 3-D objects by a gripper finger. *Robotica*, 1983, *1*, 217-222.

Klatzky, R. L., Lederman, S. J., & Metzger, V. A. Identifying objects by touch: An "expert system". *Perception & Psychophysics*, 1985, *37*, 299-302.

Lederman, S.J. & Browse, R.A. Haptic Feature Integration. submitted August 1985 to *The Journal of Experimental Psychology: General*.

Lowe, D. G. *Perceptual organization and visual recognition.* Unpublished PhD. thesis, Department of Computer Science, Stanford University, Stanford CA., 1984.

Luo, R., Tsai, W., & Lin, J. C. Object recognition with combined tactile and visual information. In A. Pugh (Ed.), *Proceedings of the 4th International Conference on Robot Vision and Sensory Controls*, London UK, October 9-11, 1984, 183-196.

Marr, D. *Vision: A computational investigation into the human representation and processing of visual information.* San Francisco: W. H. Freeman and Company, 1982.

Pugh, A. (Ed.). *Proceedings of the 4th International Conference on Robot Vision and Sensory Controls.* London UK, October 9-11, 1984.

Rose, S. A., Gottfried, A. W., & Bridger, W. H. Infants' cross-modal transfer from solid objects to their graphic representations. *Child Development*, 1983, *54*, 686-694.

Rosenfeld, A. (Ed.). *Proceedings of the Third Annual Applied Machine Vision Conference.* Chicago IL., February 27 - March 1, 1984.

Ulug, S. *Tactile feature extraction.* Unpublished MSc thesis, Department of Computing and Information Science, Queen's University, Kingston, Ontario, Canada, 1986.

Witkin, A. P., & Tenenbaum, J. M. On the role of structure in vision. In J. Beck, B. Hope, & A. Rosenfeld, (Eds.), *Human and machine vision.* New York: Academic Press, 1983.

OBSERVATIONS ON THE ROLE OF CONSTRAINTS IN PROBLEM SOLVING

Mark S. Fox

Intelligent Systems Laboratory
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

1. Introduction

Constraints have a larger role to play in heuristic search than has been demonstrated. It is possible that many of the search architecture design decisions may be deduced given a semantically complete description of the problem's constraints. The intent of the research described in this paper is to develop a semantics for the description of constraints, and a search algorithm which uses these constraints to efficiently search the combinatorial solution space.

Le concept de contrainte a un role plus important à jouer dans la recherche à base d'heuristique qu' il ne fut demontré pour l'instant. La conception d'une architecture de recherche peut souvent être déduite d'une description sémantique complète des contraintes du domaine d'application. Le but de la recherche décrite dans cet article est de développer une sémantique de description des contraintes, et un algorithme de recherche qui utilise ces contraintes pour parcourir efficacement l'espace combinatoire des solutions.

1.1. Problem-Solving Architectures

Simon (1983) has proposed that there are three "rather distinct ways ... for representing and thinking about problem solving tasks." The first views problem solving as a *search* through a model space of nodes (i.e., states) and links. The second views problem solving as *reasoning*, where new statements are deduced from a set of axioms in a formal language of logic. The third views problem solving as *constraint satisfaction*, where the incremental addition of constraints narrows down a set of objects to a subset which satisfies all the constraints. While these views are not mutually exclusive, they are viewed as being distinct. In fact, a constraint satisfying algorithm is viewed as not creating new objects, but reducing the entire space of objects to a satisficing set[1] On the other hand, search techniques, such as those used for planning, can be synthetic; incrementally constructing a solution as part of the search process.

Search, coupled with heuristics, has been the most successful of the techniques for solving real, combinatorially complex problems. Examples include:

- Hearsay II: Speech Understanding (Erman et al., 1980)
- Molgen: Molecular experiment planning (Stefik, 1981a)
- ISIS: Job-shop scheduling (Fox, 1983)
- Aladin: Aluminum alloy design (Farinacci et al., 1986)
- R1/XCON: Computer configuration (McDermott, 1983)
- GARI: Process planning (Descotte & Latombe, 1985)

Surprisingly, each of these systems use constraints, in one form or another, to guide the search process. Consequently, contraint based problem solving is less distinct than one would assume. The question is: What role do constraints actually play in search based problem solving?

1.2. Problems with Heuristic Search

The design of search architectures is an art. Skilled AI engineers approach a problem with a set of techniques, which by example, have been shown useful in other tasks. These techniques have been acquired through experience. In fact, the training of AI engineers is similar to the medival guild system where apprentices work with masters for some period of time. Only recently, have papers been prepared which bear a similarity to a guild member's handbook (Stefik et al., 1981; Kline & Dolins, 1985). My recent investigations (Fox, 1983; Baykan & Fox, 1986) in the use of constraints as the primary representational paradigm in solving problems using heuristic search has led me to conclude that constraints have a larger role to play than previously imagined. Specifically, a number of the architectural design decisions can be based upon the knowledge embedded in a sufficiently rich constraint representation. This may lead to the automation of parts of the problem solving architecture design process[2]

In the rest of this paper, I review the evolution of heuristic search and contraint based problem solving techniques. This is followed by a historical review of the ISIS job shop scheduling system, whose exploration led to many of the ideas formulated here. Lastly, observations about constraints are explored with respect to their relevance in determining the architecture of a domain specifice problem solver.

2. The Evolution of Heuristic and Constraint-Based Search

If constraints are to play a role in determining the structure of search, it is necessary to identify those structures. This section reviews the evolving set of heuristic search structures, followed by a review of the use of constraints in search.

[1] This assumes the ability to enumerate a set of objects from which to choose.

[2] At this time, I am not so bold to conclude that all of it could be automated.

2.1. Heuristic Search

Search explores a network of states in which each state represents a step along the path to a solution. The most facile use of search *anchors* it at the initial state and generates a tree in a breadth or depth first manner.

Game playing extends the concept of search to include the heuristic rating of states using domain specific knowledge. Due to the large size of the search space, game playing systems are required to prune the examined states. To achieve this, an *evaluation function* is used to rate states, in effect answering: Of all the legal moves that can be made, what are the ***preferred*** moves? A variety of search algorithms have been used such as min-max, A* (Nilsson, 1971), and B* (Berliner, 1979). An evaluation function can be used to measure structure, i.e., ply and fanout, reducing the technique to a breadth or depth first search.

The search techniques described above assume the application of all operators at each state. Means-ends analysis (Newell & Simon, 1956), provides for the selection of the *best operator* to reduce the difference between the current state and the goal state. Consequently, operators are ordered in addition to states.

Early robot planning research resulted in the formalization of operators in the predicate calculus. The STRIPS system (Fikes & Nilsson, 1971) represented operators as rules with *pre-conditions and post conditions*. GPS-like means-end analysis was used to plan tasks.

Simon (1962) recognized that a planning system in a real domain will have to struggle with the size of the search space. He proposed that search be done at differing *levels of abstraction*. By designating search hierarchies, search can proceed at the highest, least detailed level and use the results to constrain search at the next, more detailed level, and so on. One could view the ordering of differences and operators in GPS's difference table as an implicit hierarchy. The first use of this concept was in ABSTRIPS (Sacerdoti, 1974). By separating pre-condition variables into levels of importance, the pre-conditions would contain only the variables important at the current level of planning.

Another type of reasoning with differing levels of abstraction can be found in the Hearsay-II speech understanding system (Erman et al., 1980). These levels were defined by data abstractions.

Goal protection is another issue for search. The result of one action may be reversed by another before the result could be used in the overall achievement of the goal. To deal with this, the HACKER system (Sussman, 1975) used a *debugging* approach to fix a plan after it was constructed. A set of *critics* were dynamically constructed to recognize errors and suggest corrections. The NOAH system (Sacerdoti, 1975) took a *least-commitment* approach to planning. NOAH would not sequence operations unless forced. This approach reduced the amount of backtracking necessary to secure a legal plan because the current plan did not make any unnecessary sequencing decisions.

Hayes-Roth & Hayes-Roth (1980) call the combination of bidirectional problem-solving and the ability to start problem-solving at any point in the search space (island-driving as opposed to left to right) found in Hearsay-II, *opportunistic* reasoning. Opportunistic reasoning reduces the search space by focussing the planning effort in areas that are of high certainty and/or highly constrained. By extrapolating from these "islands", further constraints on the more uncertain parts of the planning space should be generated.

Problem-solving architectures were extended by incorporating the *interacting experts* paradigm. This paradigm first appeared in Hearsay-II in the form of a blackboard architecture in which the experts communicate by generating and testing hypotheses on a shared blackboard. A somewhat different idea appears in Beings (Lenat, 1975) and Actors (Hewitt, 1973) where the experts communicate directly.

Hearsay-II also introduced the concept of *focus of attention* (Hayes-Roth & Lesser, 1976). Policy modules in Hearsay-II determined the sequence of knowledge source executions. When parts of the utterance remained uninterpreted, Hearsay-II dynamically determined what parts of the search space required more attention and turned the systems resources towards reducing the uncertainty in those areas. By understanding what problem-solving methods Hearsay-II had available (i.e., knowledge sources) and its resource constraints, it would decide the best next action. The ability to reason about "how to reason" (or plan) has been called *meta-planning* in MOLGEN (Stefik, 1981b) and also appeared as *meta-rules* in TEIRESIAS (Davis, 1976; Davis & Buchanan, 1977). The implementation of meta-planning in MOLGEN used the concept of levels of representation for operators, in addition to what was commonly found for variables.

The blackboard architecture has been extended to include *multiple blackboards*, some of which are concerned with control, others which are concerned with the problem domain (Hayes-Roth & Hayes-Roth, 1980; Rychener et al., 1986).

Although much of the above search research was concerned with how to reduce the search space, other aspects of the search problem must be considered. Game playing systems introduced search techniques for adversary-oriented games. That is, the search would consider both the programs' moves and the opponents moves to determine a next move. The concept of *adversary-oriented planning* has reappeared as *counter-planning* in the POLITICS system (Carbonell, 1979). This research can be viewed as a form of goal-protection in which the system has to consider what the adversary may do to prevent the system from achieving its goals.

All of the above search research is concerned with achieving a single goal. But another type of search is concerned with the satisfaction of *multiple, possibly competing goals*. NUDGE is an early system which focused on multiple goal satisfaction (Goldstein & Robert, 1977). A heuristic approach was developed for the domain of appointment calendar maintenance. This research was unique because it included rules for the *relaxation* of constraints. When a schedule could not be found that satisfied the existing constraints, it used the rules to propose alternatives (possibilities) by relaxing certain constraints, such as preferences. In this case, the preference constraint was simply removed. Other rules peculiar to the appointment domain were used to alter existing calendar requirements until a viable schedule was produced.

Aladin, an alloy design system (Rychener et al, 1986) deals with multiple goal protection through *over satisfaction*. By over satisfying a continuous goal initially, other goals which may reduce its satisfaction will not reduce it enough to be "broken".

2.2. Search with Constraints

In parallel, a somewhat divergent set of work has led to an understanding of how to solve problems using constraints. Linear programming , at one end of the spectrum, appears to bear little relation to the AI theory of problem-solving. At the other end is the constraint-directed heuristic search of REF-ARF (Fikes, 1971) and MOLGEN (Stefik, 1981) which combines a constraint representation with heuristic search.

One of the earlier works in constraint analysis was REF-ARF (Fikes, 1970). Its task was similar to the linear programming task. Given a set of linear inequalities that restrict the possible values of a set of variables, can value assignments be found for them? Rather than a brute force search for a set of bindings that satisfied all the constraints (equations), REF-ARF used the constraints to reduce the generated binding set. Hence, the system can be viewed as a classical generate and test, by which the system was able to take the constraints and use them in the generator to reduce the size of the search space.

Another form of constraint is an adjacency network such as a grammar. A grammar defines the legal sentences that can be formed from a symbol set. The grammar can be viewed as a constraint on the symbols that will be recognized and/or generated. It defines what symbols are compatible with other symbols when linearly ordered. Another example is the conceptual hierarchy of the SEMANT knowledge source of Hearsay-II (Fox & Mostow, 1977). It is similar to a grammar, but relaxes the sequence constraint at the phrase level, allowing ungrammatical sentences, and sentence fragments to be understood. A third instance is the 3D space description network used in ARGOS (Rubin, 1978). In this case, a network was used to define adjacencies of objects in a visual scene, and used to constrain the set of acceptable labelings of an image.

In many real-world applications, constraints are not binary, but are continuous. For example, in image understanding, how a pixel is to be labeled is determined by the labels of neighboring pixels. The knowledge of how to do neighborhood based labelling is at best uncertain, hence the constraints that tie pixels together return a certainty rating for each of the possible labellings of the pixel. The higher the rating, the more probable that the label is correct. This type of constraint is the chief mechanism of *relaxation* (Zucker, 1976). Relaxation can also be viewed as a network constraint system. The goal is to assign a value to each node. A nodes value is constrained by the compatibility rules on the incident arcs. The CONSTRAINTS system (Sussman & Steele, 1981) can be viewed (loosely) as the dual of relaxation. Behavior rules are associated with nodes, and values with arcs. When an arc value changes, a node's rules determine its effect (i.e., value) on the other incident arcs. The system could recognize inconsistencies in arc values due to the lack of uncertainty in rule knowledge.

As search moved from single level to hierarchical, so has relaxation and relaxation-like processes. Single level relaxation often did not have enough information to adequately label a scene. By creating multiple levels of representation, higher levels of knowledge could be incorporated (Zucker, 1977).

The next step was to combine both binary and continuous constraints in a hierarchical system. Again, image understanding research has been the area for this research (Ballard et al., 1977; Russell, 1979). The representation of constraints in image understanding has also been extended to predicate calculus. Davis (1980) makes the case that predicate calculus is a better representation for discrete relaxation constraints.

MOLGEN combined planning with constraint-analysis (Stefik, 1981a). As plans were broken into sub-problems, variable value constraints determined in one subproblem were propagated to other subproblems. Hence, variables accumulated constraints across subproblems before an actual binding was chosen (a least commitment approach).

Engelman et al. (1980) in interactive frame instantiation associate constraints with groups of slots. An interesting feature of their approach is that constraints form buckets, each having its own priority. Hence, constraints have a priority ordering.

In planning driving paths through a town, McCalla (1978) considered constraints such as possible routes, and time and space restrictions.

Fukumori (1980) used a constraint-based approach to determine the arrival and departure times of trains at stations. Trains initially have fuzzy times assigned (i.e., a time span or *belt*). Constraints then reduce the size of the belt. The problem is much simpler than the general scheduling problem: trains had only one route and two resources, a track and stations. The fuzziness of times is similar to that used in Hearsay-II to denote the time span of an hypothesis when its boundaries were uncertain.

The GARI system (Descotte & Latombe, 1985) combines constraint satisfaction with least committment. By evaluating constraints in priority order, precendence was introduced into parallel process plans. If the problem became overconstrained, the last constraint introduced, which is the lowest priority constraint, would be relaxed.

The WRIGHT system (Baykan & Fox, 1986) approaches the problem of kitchen design as a search problem where constraints define the search operators. Each constraint has a measure of uncertainty which signifies the level of uncertainty in the search space if the constraint is satisfied. Hence, the most certain constraint is chosen to satisfy at each step, with the hope the the resultant search space will be less complex.

2.3. Topological Assumption

The principal assumption which underlies, though implicitly, the success of these search structures is that *understanding a problem's search space will enable the selection of effective and efficient search structures.* Constraints appear to participate directly in search, as evaluation functions, as operators, and in other important ways, which will be discussed in the rest of the paper.

AI shares this assumption with Operations Research (OR). An examination of mathematical programming recognizes that OR has also been pursuing the problem of how to satisfy constraints

in combinatorially complex search spaces. The Simplex and the Bell Labs algorithms for solving linear constraint problems are the result of a topological analysis of the search space. Simplex identifies that an optimal solution can be found by visiting the vertices, while the Bell Labs solution assumes a multi-dimensional topology where jumps can be made through space between points.

OR's success in mathematical programming is restricted to the special class of linear problems. For non-linear problems, OR uses heuristics to guide the search process. In fact, recent advances appear to parallel those of AI. For example, Lagrangian relaxation is a hierarchical search technique which abstracts the complex non-linear problem to a higher level linear model whose solution guides search at the next level.

The constraint perspective investigated in the rest of the paper can be viewed as being in contention with that of Lenat (1982). He believes that the structure of search is unimportant; knowledge is everything. He fails to recognize that implicit in his heuristics is knowledge of structure. Just as Lenat looks for a richness in the representation of heuristics, I am looking for a richness in the representation of constraints, and an understanding of how they impact the structures used in a problem solving architecture.

3. Role of Constraints in Job-Shop Scheduling

In 1980, I was asked to explore the application of AI techniques to a turbine component plant's job-shop scheduling problem. The primary product of the plant was steam turbine blades. A turbine blade is a complex three dimensional object produced by a sequence of forging, milling, grinding and finishing operations to tolerances of a thousandth of an inch. Thousands of different styles of blades are produced in the plant, much of them as replacements in turbines currently in service.

The plant continuously received orders for one to a thousand blades at a time. Orders fell into at least six categories:

1. Forced outages (FO): Orders to replace blades which malfunctioned during operation. It is important to ship these orders as soon as possible, no matter what the cost.

2. Critical replacement (CR) and Ship Direct (SD): Orders to replace blades during scheduled maintenance. Advance warning is provided, but the blades must arrive on time.

3. Service and shop orders (SO, SH): Orders for new turbines. Lead times of up to three years may be known.

4. Stock orders (ST): Order for blades to be placed in stock for future needs.

The portion of the plant studied has from 100 to 200 orders in process at any time.

Parts are produced according to a process routing. A routing specifies a sequence of operations on the part. An operation is an activity which defines:

- Resources required such as tools, materials, fixtures, and machines,
- Machine setup and run times, and
- Labor requirements.

In the plant, each part number has one or more process routings containing ten or more operations[3]. Process routing variations may be as simple as substituting a different machine, or as complex as changing the manufacturing process. Further more, the resources needed for an operation may also be needed by other operations in the shop.

In AI terms, job-shop scheduling is a planning problem with the following characteristics:

- It is a *time-based* planning problem (i.e., scheduling) in which activities must be selected, sequenced, and assigned resources and time of execution.

- It is a *multi-agent* planning problem. Each order represents a separate agent for which a plan/schedule is to be created. The number of agents to be scheduled is in the hundreds.

- The agents are *uncooperative*. Each is attempting to maximize its own goals.

- *Resource contention* is high, hence closely coupling decisions.

- Search is *combinatorially explosive*. 85 orders moving through ten operations without alternatives, with a single substitutable machine for each operation and no machine idle time has over 10^{880} possible schedules.

An expert systems approach was used to construct the scheduler. This approach assumed that one or more experts could be interviewed to acquire the rules which govern their decision process. During our discussions, we found that orders were not scheduled in a uniform manner. Each scheduling choice entailed side effects whose importance varied by order. One factor that continuously appeared was the reliance of the scheduler on information other than due dates, process routings, and machine availability. The types and sources of this information were found by examining the documents issued by the scheduler. A schedule was distributed to persons in each department in the plant. Each recipient could provide information which could alter the existing schedule. In support of this observation, we found that the scheduler was spending 10%-20% of his time scheduling, and 80%-90% of his time communicating with other employees to determine what additional "constraints" could affect an order's schedule. These constraints included operation precedence, operation alternatives, operation preferences, machine alternatives and preferences, tool availability, fixture availability, NC program availability, order sequencing, setup time reduction, machine breakdowns, machine capabilities, work-in-process time, due dates, start dates, shop stability, cost, quality, and personnel capabilities/availability.

From this analysis, I concluded that the object of scheduling is not only meeting due dates, but satisfying the many constraints found in various parts of the plant. Scheduling is not a distinct function, separate from the rest of the plant, but is highly

[3] Multiple process routings correspond to a network of activities, each path representing a separate plan.

connected to and dependent upon decisions being made elsewhere in the plant. The added complexity imposed by these constraints leads schedulers to produce inefficient schedules. Indicators such as high work-in-process, tardiness, and low machine utilization support this conclusion[4]. Hence, any solution to the job-shop scheduling problem must identify the set of scheduling constraints, and their affect on the scheduling process.

Once the issue of designing a constraint-directed scheduling system was identified, a decision was made to solve the problem by constructing the ISIS family of systems. The purpose of the family is to investigate the performance of successively more sophisticated search architectures (Fox & Reddy, 1981). At each stage experiments have been run to measure the effectiveness of the architecture[5]. The rest of this section describes the family of systems called ISIS.

3.1. ISIS-0

3.1.1. ISIS-0 Goals

In the fall of 1980, work began on ISIS-0. The purpose was to identify the types of constraints used by schedulers, and the extent to which they could prune the space of alternative schedules.

3.1.2. ISIS-0 System Architecture

ISIS-0 employed a simple best-first, backtracking approach using constraints as a dynamically defined evaluation function. The salient points of the search architecture include:

- Each order was scheduled separately, in priority order, as determined by a combination of order category and due date.
- Search could be performed forward from the order's start date or backward from the order's due date.
- Operators would generate alternative operations, machines, and operation times. The shop was loaded hence the availability of resources at a particular time was restricted.
- States represent partial schedules. A path through the network determines a complete schedule.
- Constraints were either imposed exogenously by the scheduling person upon the system, or were already embedded in the factory model and their applicability determined at each point in the search space.
- Propagation of constraints was performed when scheduling decisions early in the search path restricted decisions further on.

ISIS-0 was completed in December 1980 and partially demonstrated, bugs and all, at the sponsoring plant.

3.1.3. Constraint Categories

Research on version 0 of ISIS yielded five broad categories of constraints. The first category encountered is what I call an *Organizational Goal*. Part of the organization planning process is the generation of measures of how the organization is to perform. These measures act as constraints on one or more organization variables. An organizational goal constraint can be viewed as an expected value of some organization variable. For example:

- **Due Dates**: A major concern of a factory is meeting due dates. The lateness of an order affects customer satisfaction.
- **Work-In-Process**: Work-in-process (WIP) inventory levels are another concern. WIP inventory represents a substantial investment in raw materials and added value. These costs are not recoverable until delivery. Hence, reducing WIP time is desirable.
- **Resource Levels**: Another concern is maintaining adequate levels of resources necessary to sustain operations. Resources include personnel, raw materials, tools, etc. Each resource will have associated constraints. For example, labor size must be smoothed over a month's interval, or raw materials inventory may have to be limited to a two day supply.
- **Costs**: Cost reduction can be another important goal. Costs may include material costs, wages, and lost opportunity. Reducing costs may help achieve other goals such as stabilization of the work force.
- **Production Levels**: Advance planning also sets production goals for each cost center in the plant. This serves two functions: it designates the primary facilities of the plant by specifying higher production goals, and also specifies a preliminary budget by predicting how much the plant will produce. One outcome of this activity is a forecast of the work shifts that will be run in various areas of the plant.
- **Shop Stability**: Shop stability is a function of the number of revisions to a schedule and the amount of disturbance in preparation caused by these revisions. It is an artifact of the time taken to communicate change in the plant and the preparation time.

One can view all organizational goal constraints as being approximations of a simple profit constraint. The goal of an organization is to maximize profits. Scheduling decisions are then made on the basis of current and future costs incurred. For example, not meeting a due date may result in the loss of a customer and, in turn, erosion of profits. The longer the work in process time, the greater the carrying charge will be for raw materials and value-added operations. Maintaining a designated production level may distribute the cost of the capital equipment in a uniform manner. In practice, most of these costs cannot be accurately determined and must therefore be estimated.

Physical constraints determine a second category of constraint. Physical constraints specify characteristics which limit functionality. For example, the length of a milling machine's workbed may limit the types of turbine blades for which it can be used for. Similarly, there are specific machine set-up and

---

[4] It is unfair to measure a scheduler's preformance based on the above measures alone. Our analysis has shown that scheduling is a complex constraint satisfaction problem, where the above indicators illustrate only a subset of constraints that the scheduler must consider. Schedulers are expert in acquiring and "juggling" the satisfaction of constraints.

[5] Too few AI systems today attempt to measure their effectiveness.

processing times associated with different manufacturing operations.

Causal restrictions constitute a third category of constraint. They define what conditions must be satisfied before initiating an operation. Examples of causal constraints include:

- **Precedence:** A process routing is a sequence of operations. A precedence constraint on an operation states that another operation must take place before (or after) it. There may be further modifiers on the constraint in terms of minimum or maximum time between operations, product temperature to be maintained, etc.
- **Resource Requirements:** Another causal constraint is the specification of resources that must be present before or during the execution of a process. For example, a milling operation requires the presence of certain tools, an operator, fixtures, etc.

A fourth category of constraint is concerned with the *availability* of resources. As resources are assigned to specific operations during the production of a schedule, constraints declaring the resources unavailable for other uses during the relevant time periods must be generated and associated with these resources. Resource availability is also constrained by the work shifts designated in the plant, machine maintenance schedules, and other machine down times (e.g. breakdowns).

A fifth category of constraint is *preference*. A preference constraint can also be viewed as an abstraction of other types of constraints. Consider a preference for a machine. It expresses a floor supervisor's desire that one machine be used instead of another. The reason for the preference may be due to cost or quality, but sufficient information does not exist to derive actual costs. In addition to machine preferences, operation preferences, and order sequencing preferences exemplify this type of constraint.

Figure lists the variety of constraints we have identified as well as the categories we have used to classify them.

Constraint	Org. Goal	Physical	Causal	Pref.	Avail.
Operation alternatives			x		
Operation Preferences				x	
Machine alternatives			x		
Machine Preferences				x	
Machine physical constraints		x			
Set-up times	x	x			
Queue ordering preferences				x	
Queue stability	x				
Due date	x				
Work-in-process	x				
Tool requirement			x		
Material requirement			x		
Personnel requirement			x		
Resource reservations					x
Shifts	x				x
Down time					x
Productivity achieved	x				
Cost	x				
Productivity goals	x				
Quality	x	x			
Inter-operation transfer times			x		

3.1.4. Constraint-Directed Search Concepts

Constraint-Directed Evaluation. ISIS-1 dynamically constructed a different evaluation function for each state in the search space. It constructs the evaluation function out of the constraints which have been resolved to be applicable to the state under consideration. Each constraint contributed both an importance (i.e., weight) and utility. Constraints were resolved by extracting them from the resources and operations defined in the particular state.

3.2. ISIS-1

3.2.1. ISIS-1 Goals

ISIS-0 identified the broad categories of constraints but more work on representation and search architecture was required. In January 1981, work on the second version of ISIS began. The intent of this system was twofold. Given the central role of constraints to determine a job shop schedule, a major thrust of our research focused on the identification and characterization of the constraint knowledge required to support an effective constraint-directed search. Consider the imposition of a due date. In its simplest form, this constraint would be represented by a date alone, the implication being that the job be shipped on that date. In actuality, however, due dates may not always be met, and such a representation provides no information as to how to proceed in these situations. An appropriate representation must include the additional information about the due date that may be necessary in constructing a satisfactory schedule. For example:

- How important is the constraint relative to the other known constraints? Is it more important to satisfy the cost constraint than the due date?
- If I cannot find a schedule which satisfies the constraint, are there relaxations of the constraint which can be satisfied. I.e., is there another due date which is almost as good?
- If there are relaxations available for the constraint, are any more preferred? Perhaps I would rather ship the order early rather than late.
- If I chose a particular relaxation, how will it affect the other constraints I am trying to satisfy? Will meeting the due date negatively or positively affect the cost of the order?
- Under what conditions am I obliged to satisfy a constraint? What if there are two constraints specified for the same variable, i.e., two different due date for the same lot? Or there may two different due dates depending on the time of year.

In essence, a constraint is not simply a restriction on the value of a slot for example, but the aggregation of a variety of knowledge used in the reasoning process.

The second goal was to measure the effectiveness of the a modified Beam search (Lowerre, 1976) architecture which uses constraints.

3.2.2. ISIS-1 System Architecture

The salient points of the architecture include:

- Search is divided into three levels: Order selection, resource analysis, and resource assignment.

- Each level is composed of three phases: A pre-search analysis phase which constructs the problem, a search phase which solves the problem, and a post-search analysis phase which determines the acceptability of the solution. In each phase, ISIS-1 uses constraints to bound, guide, and analyze the search.
- The order selection level is responsible for selecting the next unscheduled order to be added to the existing shop schedule. Its selection is made according to a prioritization algorithm that considers order type and requested due dates. The selected order is passed to the resource analysis level for scheduling.

- The resource analysis level selects a particular routing for the order and assigns reservation time bounds to the resources required to produce it. Pre-search analysis begins with an examination of the order's constraints, resulting in the determination of the scheduling direction (either forward from the start date or backward from the due date), the creation of any missing constraints (e.g. due dates, work-in-process), and the selection of the set of search operators which will generate the search space. A beam search is then performed using the selected set of search operators. The search space to be explored is composed of states which represent partial schedules. The application of operators to states results in the creation of new states which further specify the partial schedules under development. Depending on the results of pre-search analysis, the search proceeds either forward or backward through the set of allowable routings for the order. An operator that generates states representing alternative operations initiates the search, in this case generating alternative initial (or final) operations.

  Once a state specifying an operation has been generated, other operators extend the search by creating new states which bind a machine and/or execution time to the operation. A variety of alternatives exist for each type of operator. For example, two operators have been tested for choosing the execution time of an operation. The "eager reserver" operator chooses the earliest possible reservation for the operation's required resources, and the "wait and see" operator tentatively reserves as much time as available, leaving the final decision to resource selection level. This enables the adjustment of reservations in order to reduce work-in-process time. Alternative resources (e.g. tools, materials, etc.) are generated by other operators. Each state in the search space is rated by the set of constraints found (resolved) to be relevant to the state and its ancestors. This set is determined by collecting the constraints attached to each object (e.g. machine, tool, order, etc.) specified by the state and applying resolution mechanisms. Each constraint assigns a utility between 0 and 2 to a state; zero signifies that the state is not admissible, 1 signifies indifference, 2 maximal support. The rating of a state with multiple constraints is the mean of the utilities assigned by the constituent constraints, each weighted by the the importance of the assigning constraint.

  Once a set of candidate schedules has been generated, a rule-based post search analysis examines the candidates to determine if one is acceptable (a function of the ratings assigned to the schedules during the search). If no acceptable schedules are found, then diagnosis is performed. First, the schedules are examined to determine a type of scheduling error and the appropriate repair. Intra-level repair may result in the re-instantiation of the level's search. Pre-analysis is performed again to alter the set of operators and constraints for rescheduling the order. Inter-level repair is initiated if diagnosis determines that the poor solutions were caused by constraint satisfaction decisions made at another level. Inter-level diagnosis can be performed by analyzing the interaction relations linking constraints. A poor constraint decision at a higher level can be determined by the utilities of constraints affected by it at a lower level, and an alternative value can be chosen.

  This level outputs reservation time bounds for each resource required for the operations in the chosen schedule.

- The resource selection level establishes actual reservations for the resources required by the selected operations which minimize the work-in-process time. The algorithm takes the time bounds for each resource and proceeds to shift the availability of the resoruces within the bounds so that a schedule is produced which minimizes work-in-process time.

- In addition to incrementally scheduling orders for production as they are received by the shop, the ISIS-1 search architecture could be exploited in a reactive manner. As unexpected events (e.g. machine breakdowns) cause disruptions in the existing shop schedule, ISIS-1 needed only to reschedule the affected orders. Previous reservations were transformed into preference constraints so that the new search for a schedule for the affected order would follow as much as possible to original schedule. This results in a minimal amount of change, and provides continuity in the shop schedules generated over time.

3.2.3. Constraint Representation

Let us examine the representational issues raised by these examples and, correspondingly, the salient features of the ISIS constraint representation (additional details may be found in Fox (1983) and Smith (1983)).

One of the central issues that must be addressed by the constraint representation is *conflict*. Consider cost and due-date constraints. The former may require reduction of costs while the latter may require shipping the order in a short period of time. To accomplish the latter, faster, more expensive machines may be required, thereby causing a conflict with the former. In short, it may not be possible to satisfy both constraints, in which case one or both must be relaxed. This is implicitly accomplished in

mathematical programming and decision theory by means of utility functions and the specifications of relaxation through bounds on a variable's value. In AI, bounds on a variable are typically specified by predicates (Engleman, 80; Stefik, 81) or choice sets (Steele, 80; Waltz, 75).

Given the diverse in the types of constraints present in the job shop scheduling domain, it is necessary to provide a variety of forms for specifying *relaxations* (i.e. alternative values) of constraints. Accordingly, relaxations may be defined within the ISIS constraint representation as either predicates or choice sets, which, in the latter case, are further distinguished as discrete or continuous. However, the simple specification of bounds on a variable provides no means of differentiating between the values falling within these bounds, a capability that is required by ISIS both for generating plausible alternative schedules for consideration and for effectively discriminating among alternative schedules that have been generated to resolve a given conflict. The necessary knowledge is provided by associating a *utility* with each relaxation specified in a constraint, indicative of its preference among the alternatives available. The utility of a relaxation may have more than one interpretation, which can be problematic. In the case of the due date constraint, it represents a preference for shipping on time rather than late. In the case of shifts it represents the degree of difficulty with which another should can be added. In both cases, the focus is on the difference in utility between alternative relaxations. This difference is called the *elasticity* of the relaxation. The greater the decrease in utility, the lower the elasticity. If the information were available, the utility measure would reduce to a cost function.

The relative influence to be exerted by a given constraint, i.e. its *importance*, is a second aspect of the constraint representation. Not all constraints are of equal importance. The due date constraint associated with high priority orders, for example, is likely to be more important than an operation preference constraint. Moreover, the relative importance of different types of constraints may vary from order to order. In one order, the due date may be important, and in another, cost may be important. Both of these forms of differentiation are expressible within the ISIS constraint representation; the former through the association of an absolute measure of importance with each constraint, and the latter by the use of scheduling goals which partition the constraints into importance classes and assign weights to be distributed amongst each partition's members. This knowledge enables ISIS to base its choices of which constraints to relax on the relative influence exerted by various constraints.

A third form of constraint knowledge explicitly represented is constraint *relevance*, which defines the conditions under which a constraint should be applied. Given that constraints are attached directly to the schemata, slots, and/or values they constrain, constraint relevance can be determined to a large degree by the proximity of constraints to the portion of the model currently under consideration. A finer level of discrimination is provided by associating a specific procedural test with each constraint. However, there are situations in which problems arise if the applicability of constraints is based solely on their context sensitivity to the current situation. First, many constraints tend to vary over time. The number of shifts, for example, fluctuates according to production levels set in the plant. Consequently, different variants of the same constraint type may be applicable during different periods of time. Within the ISIS constraint representation these situations are handled by associating a temporal scope with each variant, organizing the collection of variants according to the temporal relationships among them, and providing a resolution mechanism that exploits the organization. A second problem involves inconsistencies that might arise with respect to a given constraint type. Since ISIS is intended as a multiple user system, different variants of the same constraint type could quite possibly be created and attached to the same object in the model. For example, both the material and marketing departments may place different and conflicting due date constraints on the same order. In this case, a first step has been taken in exploiting an authority model of the organization to resolve such inconsistencies.

A fourth aspect of the constraint representation concerns the *interactions* amongst constraints. Constraints do not exist independently of one another, but rather the satisfaction of a given constraint will typically have a positive or negative effect on the ability to satisfy other constraints. For example, removing a machine's second shift may decrease costs but may also cause an order to miss its due date. These interdependencies are expressed as relations within the ISIS constraint representation, with an associated *sensitivity* measure indicating the extent and direction of the interaction. Knowledge of these interactions is used to diagnose the causes of unsatisfactory final solutions proposed by the system, and to suggest relaxations to related constraints which may yield better results.

A final concern is that of constraint *generation*. Many constraints are introduced dynamically as production of the schedule proceeds. For example, a decision to schedule a particular operation during a particular interval of time imposes bounds on the scheduling decisions that must be made for other operations in the production process. The dynamic creation and propagation of constraints is accomplished by attaching constraint generators to appropriate relations in the model.

Consider a constraint that restricts the length of a turbine blade that can be milled on a machine to less than 28.5 inches. This can be represented by the schema **product-length-requirement** which is a combination of a **required-constraint** and a **binary-attribute-constraint.**

product-length-requirement specifies that the foil-length of a blade is being constrained. It is obligated to being used during an airfoil-operation, and it negatively affects the airfoil-machine-preference constraint. The actual constraint is specified in **product-length-constraint**. If the constraint is satisfied then the utility returned will be 1.2, otherwise 0. The predicate of the requirement is specified by the **product-length-predicate** It specifies that any blade must have a foil-length less than 28.5 units. One potential problem in constructing this constraint is enabling the predicate to refer to slots in the root constraint (i.e., product-length-requirement). You will notice that the predicate schema is linked to the requirement schema via a PREDICATE-OF relation (inverse of predicate), and that the requirement schema is linked to the range constraint by a CONSTRAINT-OF relation (inverse of constrained-by). Each of these relations allow the inheritance of slots and values. Hence the **product-length-predicate** inherits the DOMAIN, and RELATION slots from **product-length-requirement.**

The **product-length-requirement** is not attached to the FOIL-LENGTH slot of all products, but is attached instead to the

airfoil-operation schema (i.e., contained in the **constraint** slot). It is up to ISIS to retrieve the constraint from the operation's CONSTRAINT slot, and apply its tester function to the search state

```
{{ product-length-requirement
    IS-A: range-constraint
    DURING: airfoil-operation
    CONSTRAINS: airfoil-machine-preference
        direction: neg
    DOMAIN:
        range: (type "is-a" "blade")
    RELATION: foil-length
    CONSTRAINED-BY: product-length-constraint }}
```

Figure 3-1: product-length-requirement Schema

```
{{ product-length-constraint
    CONSTRAINT-OF: product-length-requirement
    INSTANCE: required-constraint
    RELAXATION-TYPE: required
    TRUE-UTILITY: 1.2
    PREDICATE: product-length-predicate }}
```

Figure 3-2: product-length-constraint Schema

```
{{ product-length-predicate
    PREDICATE-OF: product-length-constraint
    INSTANCE: binary-attribute-predicate
    RANGE-2: 28.5
    PREDICATE: lessp }}
```

Figure 3-3: product-length-predicate Schema

and constraint. The tester retrieves the blade being scheduled and places it in the domain slot, and applies the contents of the APPLY slot in the predicate to its schema.

Another manufacturing constraint is the specification of shifts. A shift defines the time that a work center is available for work. Historically, it has been discrete, specifying one, two, or three shifts during a work day. In addition, the number of shifts on a week end may differ from that during a week day. Therefore, a shift constraint should specify what the normal available shifts are, what the relaxations are, and the period during which the shift constraint should be interpreted.

A shift specification may be specified as a **discrete-constraint**. The CONSISTENCY of the slot is exclusive, specifying that only one shift constraint may exist for the slot. No alternatives are specified at this point.

```
{{ shift-constraint
    IS-A: range-constraint
    DOMAIN:
        range: (or (TYPE is-a machine) (TYPE is-a work-center))
    RELATION: shift }}
```

Figure 3-4: shift-constraint Schema

```
{{ shift-constraint-spec
    is-a: discrete-constraint
    CONSISTENCY: exclusive }}
```

Figure 3-5: shift-constraint-spec Schema

```
{{ shift
    START-TIME:
    END-TIME:
    WORK-WEEK: }}
```

Figure 3-6: shift Schema

An example of a shift constraint is that specified for a wmf1 machine

```
{{ wmf1-shift
    IS-A: shift-constraint
    DOMAIN: wmf1
    RELATION: shift
    CONSTRAINED-BY: wmf1-shift-constraint }}
```

Figure 3-7: wmf1-shift Schema

The range constraint specifies the domain of the constraint and relation. That is, the constraint affects the SHIFT slot of the wmf1. The contents of the CONSTRAINED-BY slot is the name of the constraint: **wmf1-shift-constraint**. It describes a start-time, end-time, and day for the shift.

The shift constraint is not a schema constraint. Each relaxation completely specifies the start-time, end-time, and work week. They cannot be relaxed individually. The contents of the RELAXATION slot specify another shift, **wmf1-shift-relaxation**, to be used in addition to the first constraint (the DISCRETE-TYPE of the constraint is inclusive).

The constraint is interpreted by taking the value of the TESTER slot from **wmf1-shift** (not shown) and applying it to the pair (<state> **wmf1-shift**). The tester will retrieve the discrete constraint and find the value which matches the value under

```
{{ wmf1-shift-constraint
    INSTANCE: shift-constraint-spec
    RELAXATION-VALUE: {{ INSTANCE shift
                         START-TIME:    8:00
                         END-TIME:      16:00
                         WORK-WEEK:     (OR monday tuesday wed
                                         thursday friday) }}
    RELAXATION-UTILITY:  2
    RELAXATION: wmf1-shift-relaxation }}
```

Figure 3-8: wmf1-shift-constraint Schema

consideration (i.e., specified inthe state) and return the relaxation utility.

```
{{ wmf1-shift-relaxation
    INSTANCE: shift-constraint
    RELAXATION-VALUE: {{ INSTANCE shift
                         START-TIME:    16:00
                         END-TIME:      24:00
                         WORK-WEEK:     (OR monday tuesday wed
                                          thursday friday)   }}
    RELAXATION-UTILITY:   1.2
    DISCRETE-TYPE:        inclusive   }}
```

Figure 3-9: wmf1-shift-relaxation Schema

The basic due-date-constraint is a continuous value constraint which constrains the due-date slot of a lot. The choice of a due-date has a utility specified by the PIECE-WISE-LINEAR-UTILITY. The utility is specified by (shipping-lateness utility) pairs. An example of its use is a due date for forced outage orders. The tester for **due-date-constraints** takes the search state and the constraint as parameters, retrieves the due date being considered int the state, or predicts one, and applies the value of the utility function slot to the due date. The utility function uses the PIECE-WISE-LINER-UTILITY value to interpolate and return a utility.

fo-due-date specifies that the utility of the due date chosen is 2 if it less than or equal to the the requested due date. It is linearly decreasing to 0.2 if it greater than 0 days late and less than 7. And is 0.2 if greater than 7 days late.

```
{{ due-date-constraint
    IS-A: range-constraint
    DOMAIN:
        range: (type "is-a" "lot")
    RELATION: due-date
    CONSTRAINED-BY:
        range: (type "is-a" "due-date-constraint")
    TESTER: due-date-tester
    PRIORITY-CLASS:   }}
```

Figure 3-10: due-date-constraint Schema

```
{{ due-date-constraint-spec
    IS-A: continuous-constraint
    CONSISTENCY: exclusive
    UTILITY-FUNCTION: interpolate
    PIECE-WISE-LINEAR-UTILITY: }}
```

Figure 3-11: due-date-constraint-spec Schema

```
{{ fo-due-date
    IS-A: due-date-constraint
    PRIORITY-CLASS: forced-outage
    CONSTRAINED-BY: {{ INSTANCE due-date-constraint
                       PIECE-WISE-LINEAR-UTILITY: ( (0 2) (7 0.2)
    }}
```

3.2.4. Performance of ISIS-1

Experiments were performed with a real plant model and order data. In each experiment, an empty job shop was loaded with a representative set of 85 orders with arrival times distributed over a period of two years. The various types of constraint knowledge influencing the development of schedules in these experiments included alternative operations, alternative machines, requested due dates, requested start dates, operation time bounds, order priority classification (with orders falling into 4 priority classes), work-in-process restrictions, queue ordering constraints to reduce setup time, machine constraints on product form and length, resource availability, and shop stability (minimizing pre-emption).

A number of experiments were performed. These experiments explored the effects of alternative constraints, alternative search operators, and beam width size. A detailed discussion of all experiments may be found in Fox (1983).

The gantt chart[6] shown in **Figure 3-1** depicts a schedule generated by ISIS-1.

Figure 3-1: Version 1 Gantt Chart

The schedule is a poor one; 65 of the 85 orders scheduled were tardy. To compound the problem, order tardiness led to high work-in-process times (an average of 305.15 days) with an overall makespan[7] of 857.4 days. The reason for these results stems from the inability of the beam search to anticipate the bottleneck in the "final straightening area" of the plant (the fts* machine on the gantt chart in Figure 3-1) during the early stages of its search. Had the bottleneck operation been known in advance, orders could have been started closer to the time they were received by the plant and scheduled earlier through the bottleneck operation.

Beam search sizes between 5 and 20 were tested. Sizes greater than 10 had little affect on the outcome, while sizes less than 10 performed more poorly.

3.2.5. Constraint-Directed Search Concepts

Constraints as Generators. Constraints which specify precedence between operations and requirements for resources can be interpreted as search operators. For example, a constraint which specifies that drilling must following milling can be interpreted as operator which extends a state for which milling is defined to be the operation into a new state for which drilling is the successor operation. Each constraint in ISIS-1 has code which interprets the constraint as an operator to be used in search.

[6] Each row represents a machine, and each column a week. If a position in the gantt chart is empty, then the machine is idle for that week. If a position contains an "o", then it is utilized for less than 50% of of its capacity. If the position contains a "@", then over 50% of its capacity is utilized. Machines that are encountered earlier in the process routings appear closer to the top of the chart.

[7] Makespan is the time taken to complete all orders.

ISIS-1's presearch analysis selects the operators from a subset of available constraints.

Constraints Bound the Search Space. The omission of constraints by pre-search analysis (e.g., alternative shifts), when defining operators, results in a bounding of the search space. This restriction on the size of the search space is intentional but can be relaxed by post-search analysis.

Generative Constraint Relaxation. The joint satisfaction of all constraints simultaneously at one time is impossible due to conflict among constraints. Relaxation is the process by which alternative solutions are explored by relaxing the specification of the constraints. Consequently, it provides a satisficing approach to constraint satisfaction. Generative relaxation is one type of relaxation process. It is a process by which alternative solutions are generated during the search process. This is accomplished by extending the code which interprets a constraint as an operator so that it uses the specified relaxations to generate alternative successor states which define alternative bindings of variables. In some cases, the number of relaxations are large (e.g., a continuous constraint such as start time of an operation), requiring the code to use a relaxation's utility to determine whether it is good enough to be generated.

Constraint Resolution and Dynamic Evaluation. ISIS-1 extends the concept of dynamic evaluation function construction by utilizing a more sophisticated form of constraint resolution.

Local Resolution. ISIS-2 dynamically resolves the set of applicable constraints at each search state. Resolution is performed by examining each schema (i.e., operation, machine, etc.) in the current state description. The contents of any CONSTRAINT slots, or constraints attached to any slots which enable the schema are added to the local resolution set. Constraints may originate from four sources:

Model-Based: Constraints may be embedded in any resource or activity in the factory model. For example, there may be physical constraints associated with a machine, sequencing constraints associated with an operation, queue ordering constraints associated with certain work centers.

Lateral Imposition: Constraints can also be propagated laterally during the search. A decision made earlier in the elaboration of a schedule may result in a constraint being attached to the lot that restricts a choice point further on in the search.

Exogenous Imposition: The user may also create and implant constraints. These constraints can be attached to anywhere in the model, or be globally attached so that it is considered at each search state.

Global Resolution. The rating of a state is a rating of the partial schedule up to the current state, and not the single choice represented by the state. Hence, the rating of a state must include not only the local constraints but the constraints applied to all the states along the partial schedule ending at the current state. Not all the constraints locally resolved at each state along the path are globally resolved. Consider the **due-date-constraint** (figure 5-10). It is a classic evaluation function as defined in heuristic search. Part of the constraint calculates the work-in-process time of the lot to the current state, and the other part *predicts* the remaining work-in-process time to the end state. Each time the constraint is applied, it is a better estimator of the work-in-process time, and should overide applications of the same constraint earlier in the partial schedule. On the other hand, the **queue-stability** constraint is applied at each state which binds a queue position. It rates the state by how much it destabilizes existing queue reservations. The greater the destabilization, the lower the rating. This constraint measures a decision made at that state, and remains invariant over future states, since any future states cannot affect an earlier state.

Constraints are classified into two categories: *invariant* and *transient*. All invariant constraints participate in the globally resolved constraint set, and only the most recent version of transient constraints participate.

Relative Resolution. All constraints are not created equal. Relative resolution differentially interprets the resolved constraints by partitioning the constraint set according to the applicable scheduling goal. A scheduling goal partitions the constraint set and defines an importance for each partition. The importance is then uniformly divided amongst the constraints in the partition.

Analytic Relaxation via Constraint Diagnosis and Repair. The completion of the beam search may result in schedules which are not acceptable due to the poor satisfaction of many of its important constraints. Analytic relaxation is defined to be the process by which the results of the search are examined to determine which "peep hole" repair of a constraint will generate a significant increase it the overall constraint rating of a schedule. In addition to the procedural embedding of situational knowledge in the form of rules (e.g., IF you cannot meet the due date THEN relax the start date constraint by starting earlier), a declarative approach was taken. Each constraint may have a **constrains** relation which links it to another constraint. If the first constraint was not acceptably satisfied (e.g., due date), then by searching along the constrains relation another constraint could be found (e.g., shifts) whose further relaxation or strengthening could impact the first constraint. Consequently, post-analysis could suggest the increase in number of shifts to pre-search analysis and have the search re-run.

3.3. ISIS-2

3.3.1. ISIS-2 Goals

ISIS-1 identified the representational requirements of constraints, and their use in directing search. Neither changes in beam width, nor alterations to existing constraints were able to significantly affect the degree to which due date and work in process constraints were unsatisfied. The cause of this problem lay with the combinatorics of the search space combined with the horizon effect. ISIS-2 was designed reduce the impact the horizon effect has on the quality of the schedules.

3.3.2. ISIS-2 Architecture

ISIS-2 constructs schedules by performing a hierarchical, constraint-directed search in the space of alternative schedules. An additional level was added between order selection and

resource analysis: capacity analysis. The purpose of this level was to consider a subset of the more important constraints in order to "look ahead" so that capacity bottlenecks could be identified in a smaller search space.

Capacity analysis takes as input the selected order from the order selection level and uses the following subset of constraints in its search: due date, start date, operation precedence and alternatives, machine requirements, and machine reservations. All other constraints are ignored. The capacity analysis level performs a dynamic programming analysis of the plant based on current capacity constraints. It determines the earliest start time and latest finish time for each operation of the selected order, as bounded by the order's start and due date. The times generated at this level are codified as operation time bound constraints which hierarchically propagated to the resource analysis level.

3.3.3. ISIS-2 Performance

ISIS-2's inclusion of a level of abstraction in the top down search hierarchy had a significant impact on the results, evidenced by the increased satisfaction of the due date constraints.

The average utility assigned by the due date constraint to lower priority "service orders", for example, almost doubled, rising from a value of 0.46 in the first experiment to a value of 0.80. The total number of tardy orders was reduced to 14. Moreover, a much lower average work-in-process time of 186.73 days was

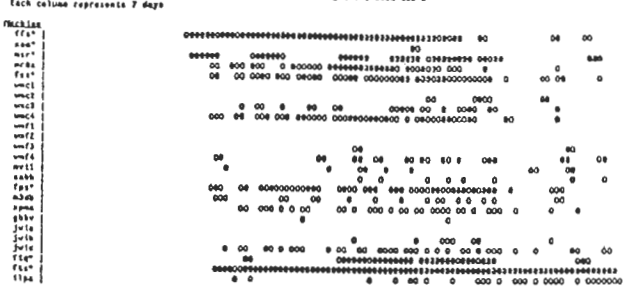


Figure 3-2: Version 7 Gantt Chart

achieved, resulting in an overall makespan of 583.25 days. In this case, inadequate machine capacity in the "final straightening area" (fts*) appeared to be the principal limitation affecting order tardiness.

3.3.4. Constraint-Directed Search Concepts

Periscoping. The improved performance of ISIS-2 rests on the ability of the capacity analysis level to identify bottlenecks and encode their effect in the form of operation time bound constraints. At the resource analysis level, whenever alternatives are generated for the time to perform a particular operation, the operation time bound constraint is resolved and evaluated. The effect is what I call *periscoping*. It is as if the evaluation of the state looked "up above" the local situation to see what problems lay further down the search path it has yet to explore. If there was a bottleneck, the operation time bound constraint would lower the utility of times which do not provide enough time to get through the bottleneck.

Constraint-Directed Focus of Attention. The hierarchical imposition of constraints of one level onto the next results in the lower level's focusing of its search on the "better" parts of the search space; reducing the complexity of the search while increasing the utility of the outcome.

Constraint Stratification. Constraints appear to fall naturally into a partial ordering in this domain according to the degree of difficulty with which they can be relaxed. For example, it is easier to alter a due date by a day than it is to add another shift. Consquently, a level of the search hierarchy, in addition to constraining the search of the next level via periscoping, can determine the values of a subset of constraints which are more difficult to change than constraints at a lower level. More on this concept will appear in the next section.

Interlevel Analytic Relaxation. The concept of analytic relaxation is extended to work across levels. If a constraint is identified as needing to be relaxed, and it is bound at a higher level, then post-search analysis will re-invoke the higher level. The level will either alter the constraint and/or re-perform the search at that level.

3.4. ISIS-3/OPIS-0

Work began on ISIS-3 (aka OPIS-0) during the summer of 1984. Though ISIS-2 made significant headway in satisfying its constraints in the presence of a high degree of resource contention, it was still believed that better use of the resources could be made resulting in higher constraint satisfaction.

3.4.1. ISIS-3 Goals

The goal of ISIS-3 was to explore the problem of varying perspectives on scheduling (Smith & Ow, 1985). In particular, the high degree of resource contention in multi-agent planning/scheduling forces one to consider scheduling the activities of the resource (i.e., machine) as opposed to the agent (i.e., order). This differs from approaches to problems for which the number of agents are small and resource contention low (e.g., (Konolige & Nilsson, 1980)).

3.4.2. ISIS-3 Architecture

The approach was to mix order scheduling with resource scheduling. (See Smith & Ow (1985) for more details.) This was accomplished as follows:

- The order selection and capacity analysis levels were merged into a single level. This new capacity analysis level used a dispatch rule simulation approach to scheduling all of the orders in parallel in the presence of the same subset of constraints associated with this level.

- The schedule at the capacity analysis level was examined for bottlenecks. Each bottleneck was then scheduled (usually one) resulting in a time at which each order which flows through the bottleneck is to be worked on.

- The bottlenecks and the schedules of orders through them were passed down to the resource analysis level. This level was modified to perform "island driving", similar to that found in Hearsay-II (Erman et al., 1980). Each bottleneck was designated an "island", and the highest priority order was selected and scheduled out (forward and backward) from the island using the original beam search with the added

constraints of this level.

- The rest of the ISIS-2 architecture remained the same.

3.4.3. ISIS-3 Performance

New experiments were performed comparing ISIS-2, ISIS-3 and the COVERT dispatch rule (Ow, 1986). In all cases, ISIS-3 outperformed the other two. The following are some significant measures:

System	
ISIS-3	
ISIS-2	
COVERT	

3.4.4. Constraint-Directed Reasoning Concepts

Islands of Certainty: Constraint-directed Focus of Attention. The bottleneck schedule produced by capacity analysis is actually a set of constraints on the search to be performed at the resource analysis level (i.e., each reservation for the bottleneck by an order is an availability constraint). Search at the resource analysis level can identify islands of certainty by the importance and utililty of the constraints. Consequently, by working on each order in priority order, the resource analysis level is able to identify, for that order, the islands of certainty in its search space (i.e., bottleneck reservations) and perform the beam search outward from those islands, resulting in "island driving".

3.5. OPIS-1

The ISIS-3 architecture is still hardwired in the sense that it performs a resource centered analysis at the capacity and level and then an order centered anlaysis at the resource analysis level. Depending on the state of the factory, one of the perspectives may be unnecessary. In the summer of 1985 work began on OPIS, the beginning of a new series of planning/scheduling systems in which opportunism in search places a greater role. In particular, the first version of OPIS, focuses on opportunistic selection of the scheduling perspective. Its architecture bears many similarities to Hearsay-II.

4. Summary

This section has provided an evolutionary view of the ISIS family of job-shop scheduling systems. Two significant results appear during this evolution. First, the development of a semantics for the representation of constraint knowledge, focusing on what is constrained, relaxation, utility, elasticity, importance, interactions and relevance. Secondly, the novel use of constraints in state generation, search space bounding, generative relaxation, resolution, analytic relaxation, periscoping, focus of attention, and stratefication.

5. Role of Constraints in Problem Solving

This section returns to the original hypothesis, that the development of an adequate semantics of constraints will lead to a better understanding of how to define the structure of search. This section ties the semantics of constraints developed in the ISIS family to the search structures of section two. In particular, a number of observations of how the semantics of constraints relates to search structures are explained.

The first five observations define the basic search architecture of states and operators.

Observation 1: "Constraints define the parameters of states in the search space."

Constraints provide a state-space view of problem solving in that they define the variables to be bound by the search process. Examples include:

- due date constrains the date shipped
- shifts constraint constrains the shifts available
- next operation constraints the current operation
- keep cost under x constrains cost of manufacturing

Observation 2: "Constraints define single state generating operators."

The specification that an attribute or relation should be restricted to a single value enables the construction of generators or search operators which will generate a state with the variable being bound to the single value.

For example, if the constraint specifies that the next operation after milling is drilling then an operator can be generated whose action is the generation of a state which binds the operation to drilling when the preceding operation is milling. These leads into the third observation:

Observation 3: "Constraints define the situation or condition of an operator."

Knowledge of constraint *relevance* determines the situation in which the constraint is to be applied. It is straightforward how the relevance knowledge could be transformed into an operator's condition. For example, a third shift may only be available during monday through friday. This condition would be encoded as an operator's condition.

Observation 4: "Complex operators are the combination of two or more constraints."

(This is another version of the question of how to moves tests into a generator.) Within ISIS are two alternative operators which choose a time at which an operation is to be performed, once the operation and machine are bound. These operators are complex; juggling concerns such as setup time reduction, not letting the operation be performed too late, order priority, shop stability, etc. The hand crafting of such an operator can be viewed as the combining of two or more constraints:

- Setup sequencing.
- Shop stability.
- Work in process.
- Order priority.

Observation 5: "Constraints define the evaluation function."

The utility associated with each constraint relaxation, coupled with a constraint's importance provides the basis for an

evaluation function. In particular, they define a linear function which is the weighted average of the utilities of all resolved constraints.

The next three observations focus on the definition of levels within a hierarchical search space.

Observation 6: "Levels of representation are defined by constrained variables part-of hierarchies."

Many variables whose values are constrained participate in part-of hierarchies. For example, the milling machine is part of the milling machine work center, and a day is part of a week. These hierarchies define levels of abstraction for each variable. In the case of scheduling, capacity analysis can be performed using machines, work centers, plants, etc., or time decisions can be made by the hour. date, week, etc.

Observation 7: "Levels of search are defined by the importance of a constraint."

The *importance* of a constraint can be used to determine which variables are to be bound first in a manner similar to that of ABSTRIPS.

Observation 8: "Levels of search are defined by the elasticity of a constraint."

Though a constraint can be relaxed, it may be difficult to do so. For example, it may be easier to ship an order two days later (i.e., relax the due date constraint) than it is to put a third shift on over the weekend. The *elasticity* of a constraint defines another stratification of the search space.

Observation 9: "Levels of search are defined by constraint interactions."

The interdependence of constraints define an interaction hierarchy. For example, the number of shifts available indirectly affect due date and work in process constraints, but not vice versa. One stratefication of the search space would have shift decisions being made at a higher level.

The next two constraints deal with issues of focus of attention.

Observation 10: "Constraints focus attention on islands of cerntainty."

Highly important constraints with low elasticity define decisions which have to conform to the constraint. These constraints define islands of certainty in the search space from which search is to be initiated. For example, if a constraint specifies that an order is to be delivered today and it is the most important constraint without any relaxations, then search begins with that order at the last operation being completed today.

Observation 11: "Constraints direct the diagnosis and repair of poor search decisions."

Diagnosis identifies poor search decisions and repair attempts a correction. In this case, the low utility of a constraint signals a problem, and a constraint's *interaction* with another constraint points to a possible peep hole optimization. Using shifts and due date constraints again, a low due date utility could be corrected by altering the shift decision.

6. Conclusion

It has been the intent of this paper to elucidate the embryo of a theory which unifies constraints with heuristic search. The theory suggests that constraints play an important role in search. That they define much of the structure of the system architecture, for which until now only heuristics existed. As of yet, the theory is incomplete; it is composed of 11 observations. Further work awaits in the elaboration of constraint semantics and the development of an interpreter which will solve a problem given a complete constraint set.

Acknowledgements

The work described in this paper represents the contributions of many people over the years. First and foremost, Steve Smith has contributed an enormous amount to the success of the project, and continues to do so as leader of the OPIS project. Portions of section 3 are based joint writings. Other notable contributers include Brad Allen, Bruce McClaren, Tom Morton, Linda Quarrie, Peng Si Ow, and Gary Strohm.

This research was supported, in part, by the Air Force Office of Scientific Research under contract F49620-82-K0017, and the Westinghouse Electric Corporation.

References

Ballard D.H., C.M. Brown, and J.A. Feldman, (1977), "An Approach to Knowledge-Directed Image Analysis", *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge MA.

Baykan C.A., and M.S. Fox, (1986), "Facility Design: An Investigation into Opportunistic Constraint Satisfaction", Technical Report, Robotics Institute, Carnegie-Mellon University, Pittsburgh PA, in preparation.

Berliner H.J., (1979), "The B* Tree Search Algorithm: A Best First Proof Procedure", *Artificial Intelligence*, Vol. 12.

Carbonell J.G., (1979), "The Counterplanning Process: A Model of Decision Making in Adverse Situations", Technical Report, Computer Science Department, Carnegie-Mellon University, February 1979.

Davis L., (1980), "A Logic Model for Constraint Propagation", Technical Report TR-137, Computer Sciences Dept., University of Texas, Austin Texas.

Davis R., (1976), "Applications of Meta Level Knowledge to the Construction, Maintenance, and Use of Large Knowledge Bases", Ph.D. Thesis, Computer Science Department, Stanford University, Report No. STAN-CS-76-552.

Davis R. and B. Buchanan, (1977), "Meta-level Knowledge: Overview and Applications," *Fifth International Joint Conference on Artificial Intelligence*, Cambridge MA, August 1977.

Descotte Y., and J-C Latombe, (1985), "Making Compromises among Antagonist Constraints in a Planner", *Artificial Intelligence*, Vol 27, pp. 183-217.

Engleman C., E. Scarl, and C. Berg, (1980), "Interactive Frame Instantiation", *Proceedings of the American Association for Artificial Intelligence*, pp. 184-186, Stanford University.

Erman L.D., F. Hayes-Roth, V.R. Lesser, and D.R. Reddy, (1980), "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty", *Computing Surveys*, Vol. 12, No. 2, June 1980, pp. 213-253.

Farinacci M.L., M.S. Fox, I. Hulthage, M.D. Rychener, (1986), "The Development of ALADIN, An Expert System for Aluminum Alloy Design", *Artificial Intelligence in Manufacturing*, Thomas Bernold, (Ed.), Springer-Verlag, to appear.

Fikes R.E., (1970), "REF-ARF: A System for Solving Problems Stated as Procedures", *Artificial Intelligence*, Vol. 1, pp. 27-120.

Fikes R.E., and N.J. Nilsson, (1971), "Strips: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, Vol. 2,pp 189-208.

Fox M.S., (1983), "Constraint-Directed Search: A Case Study of Job-Shop Scheduling", (Ph.D. Thesis), Technical Report, Computer Science Dept., Carnegie-Mellon University, Pittsburgh PA.

Fox M.S. and D.J. Mostow, (1977), "Maximal Consistent Interpretations of Errorful Data In Hierarchically Modelled Domains", *Fifth International Joint Conference on Artificial Intelligence*, Cambridge MA, 1977.

Fox M.S., and D.R. Reddy, (1981), "Constraint-Based Scheduling in an Intelligent Logistics Support System: An Artificial Intelligence Approach", Proposal to the Air Force Office of Scientific Research, August 1981.

Fukumori K., (1980), "Fundamental Scheme for Train Scheduling", MIT AI Memo No. 596, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge MA.

Goldstein I.P., and R.B. Robert, (1977), "NUDGE: A Knowledge-Based Scheduling Program", MIT AI Memo 405, Cambridge MA.

Hayes-Roth B., and F. Hayes-Roth, (1979), "A Cognitive Model of Planning", *Cognitive Science*, Vol. 3, No. 4, pp. 275-310.

Hayes-Roth F., and V.R. Lesser, (1976), "Focus of attention in a distributed logic speech understanding system", *Proceedings of the 1976 IEEE International Conference on Acoustics, Speech and Signal Processing*, Philadelphia, 1976, 416-420.

Hewitt C., (1973), The ACTOR Formalism. *Third International Joint Conference on Artificial Intelligence*, Stanford, CA.

Kline P.J., and S.B. Dolins, (1985), "Choosing Architectures for Expert Systems", Technial Report RADC-TR-85-192, Rome Air Development Center, Griffiss AFB, NY. (prepared by Texas Instruments Inc.)

Konolige K., and N.J. Nilsson, (1980), "Multiple-Agent Planning Systems", *Proceedings of the First Annual Conference of the Amercian Association for Artificial Intelligence*, August 1980, pp. 138-141.

Lenat D., (1975),"Beings: Knowledge as Interacting Experts", *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tiblisi, USSR.

Lenat D.B., (1982), "The Nature of Heuristics", *Artificial Intelligence*, Vol 19, pp. 189-249.

Lowerre B., (1976), "The HARPY Speech Recognition System", (Ph.D. Thesis), Tech. Rep., Computer Science Dept., Carnegie-Mellon University, Pittsburgh PA.

McCalla G., P. Schneder, R. Cohen, and H. Levesque, (1978), "Investigations into Planning and Executing in an Independent and Continuously Changing Microworld", AI Memo 78-2, Dept. of Computer Science, University of Toronto.

Newell A., and H.A. Simon, (1956), "The Logic Theory Machine: A Complex Information Processing System", *IRE Transactions on Information Theory*, Vol. IT-2, No. 3, pp. 61-79.

Nilsson N.J., (1971), *Problem-Solving Methods in Artificial Intelligence*, New York, N.Y.: McGraw-Hill.

Rubin S., (1978), "The ARGOS Image Understanding System", Ph.D. Thesis, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.

Russell D.M., (1979), "Where Do I Look Now?: Modelling and Inferring Object Locations by Constraints", Technical Report, Computer Science

Department, University of Rochester, Rochester NY.

Rychener M., M. Farinacci, I. Hulthage, and M.S. Fox, (1986), "Constraint-Directed, Hierarchical Search in Aladin, An Alloy Design System", Technical Report, Robotics Institute, Carnegie-Mellon University, Pittsburgh PA, in preparation.

Sacerdoti, E.D., (1974), "Planning in a Hierarchy of Abstraction Spaces", *Artificial Intelligence*, Vol 5, no.2.

Sacerdoti E.D., (1975), "A Structure for Plans as Behavior", (Ph.D. Thesis), Computer Science Dept., Stanford University.

Simon H.A., (1962), "Scientific Discovery and and the Psychology of Problem Solving", *Minds and Cosmos*, Kolodny (Ed.).

Smith S.F., (1983), "Exploiting Temporal Knowledge to Organize Constraints" Technical Report, Robotics Institute, Carnegie-Mellon University, Pittsburgh PA.

Smith S.F., and P.S. Ow, (1985), "The Use of Multiple Problem Decompositions in Time Constrainted Planning Tasks", *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 18-23 August 1985, pp. 1013-1015, Los Altos CA: Morgan Kauffman Pub Inc.

Stefik M., (1981a), "Planning with Constraints (MOLGEN: Part 1)", *Artificial Intelligence*, Vol. 16, pp. 111-140.

Stefik M., (1981b), "Planning and Meta-Planning (MOLGEN: Part 2)", *Artificial Intelligence*, Vol. 16, pp. 141-170.

Stefik M., J. Aikins, R. Balzer, J. Benoit, L. Birnbaum, F. Hayes-Roth and E. Sacerdoti, (1982), "The Organization of Expert Systems, A Tutorial", *Artificial Intelligence*, Vol 18, pp. 135-174.

Steele G.L., (1980), "The Definition and Implementation of a Computer Programming Language Based on Constraints", (PhD Thesis), MIT tech. rep. AI-TR-595, Cambridge MA.

Sussman G., (1975), *A Computational Model of Skill Acquisition*, New York: American Elsevier.

Waltz D., (1975), "Understanding Line Drawings of Scenes with Shadows", in P.H. Winston (Ed.), *The Psychology of Computer Vision*, New York N.Y.: McGraw-Hill.

Zucker S.W., (1976), "Relaxation Labelling and the Reduction of Local Ambiguities", In *Pattern Recognition and Artificial Intelligence*, C.H. Chen (Ed.), New York: Academic Press.

Zucker S.W., (1977), "Vertical And Horizontal Processes in Low Level Vision", Report No. 77-4, Electrical Engineering Dept., McGill University, Montreal, Quebec.

RULE INTERACTION IN EXPERT SYSTEM KNOWLEDGE BASES

Stan Raatz
Dept. of Computer and Information Science
University of Pennsylvania
Phila., PA 19087

George Drastal
Lab for Computer Science Research
Rutgers University
New Brunswick, NJ 08903

Abstract

The goal of this work is a system that warns the knowledge engineer about the global consequences of adding a rule to a knowledge base. To affect this, a technique is presented that identifies relationships or interactions between *sets*, and not just *pairs*, of rules, and an analysis is given which relates these interactions to the integrity of the knowledge base. We argue that the method is efficient given some reasonable assumptions, present some patterns that result from the analysis of these interactions, and discuss why this information is of interest to the knowledge engineer.

1 Introduction

This work originates from the building of a fault diagnosis expert system using the the shell *EXPERT* [11] for a large multi-component piece of electronic equipment. As the size and complexity of the knowledge base in this project increased (now over 2000 rules and 2700 observations and hypotheses), it became increasingly difficult to add a new rule with an understanding of its interactions with previous rules. This is a problem well known in the field, and we compare the approach taken here with previous work.

To illustrate such an interaction in a common sense domain, consider the following situation.

Example 1.1

Let the knowledge base contains the following rules:

1. *If* Fever *or* Chills *Then* Infection
2. *If* Cough *and* Infection *Then* Bronchitis with belief .7

Consider adding the following rule:

3. *If* Cough *Then* Bronchitis with belief .8

The addition of rule 3 renders the pair of rules 1 and 2 useless (with respect to establishing the hypothesis Bronchitis). Any time Cough is known, rule 3 establishes Bronchitis with belief .8. The pair 1 and 2 require additional observations to establish Bronchitis with a lower belief. Now consider adding the following rule:

4. *If* Cough *and* Fever *and* Shortness-of-Breath *Then* Bronchitis with belief .6

This rule will not use the observation Shortness-of-Breath. Any time the observations Cough and Fever are known, the pair 1 and 2 will establish Bronchitis with belief .7. It does not make sense to then ask for the observation Shortness-of-Breath, since the new rule cannot increase the belief in Bronchitis. Knowing Shortness-of-Breath without knowing Cough and Fever does not establish any belief.

These examples are very simple, and we would not expect a practicing knowledge engineer to make mistakes of this order. But large "real-world" knowledge bases can have obscure, subtle, and exasperating interactions that can be time consuming to identify, and may go undetected.

Previous work in this area includes the *TEIRESIAS* system [2], which allows the user to ferret out errors in the knowledge base and, using meta-rules, suggests corrections to these errors based on similarities to other rules in the knowledge base. However, the help occurs in the setting of a consultation session, and no analysis of the rules occurs at the time they were added to the knowledge base. The *EMYCIN* system [9,10], in addition to extending *TEIRESIAS*, corrects lexical errors and points out interactions between *pairs* of rules at the time of their addition. Examples of such interactions include instances of redundancy, subsumption, and inconsistency. However, it does not consider interactions between sets of rules. Neither does the recent *CHECK* program [13], which mimimally extends the above work by checking for two additional relationships between pairs of rules. The Semantic Matcher in Reboh [7,8] is similar to the *CHECK* program and the *EMYCIN* work, in that it finds pairwise relationships in the *PROSPECTOR* environment. The *SEEK* system [5], which does incorporate a concept of interaction between

sets of rules, applies only to consultation systems that have a large database of cases which can be empirically analyzed.

The technique discussed here is very simple to implement and can be applied to any expert system rule language with propositional semantics (which includes virtually all the well known ones).

2 Preliminaries

We will describe the method reported here in an abstract language that is derived from the *EXPERT* expert system shell developed at Rutgers University for designing and applying rule-based consultation models [11]. An *EXPERT* model consists of sets of hypotheses, findings (observables), and rules which relate the collection of evidence in the form of findings to the classification of the situation, i.e. the selection of a hypothesis. A finding is abbreviated by f_i, a hypothesis by h_i, and the rules used will be of the form

1. fh, which relate findings to a hypothesis. Example: X_1 & ... & $X_n \rightarrow h_i$, where $X_i = f_j$ or $X_i = [n : f_1, \ldots, f_k]$, which is interpreted as true if n of the k findings are known.
2. hh, which relate hypotheses to a hypothesis. Example: X_1 & ... & $X_n \rightarrow h_i$, $X_i = f_j$, $X_i = h_k$, or $X_i = [n : Y_1, \ldots, Y_k]$, and Y_i is any f_j or h_k.

The *EXPERT* shell has been used successfully in numerous expert system projects and has been shown to be well suited to expressing problems with classification characteristics [12].

We will use the following terminology in this paper. An *inference graph* G is the directed acyclic graph implied by a knowledge base, where nodes represent findings or hypotheses, and arcs represent rule implications. We will adopt the usual graph-related terminology, and in particular, denote the relation node n_1 is subsumed by node n_2 in G as $n_1 \leq n_2$. An *ordered set of rules* $S = \langle R_1 \ldots R_n \rangle$ is a subgraph of an inference graph along with an ordering on the rules in this subgraph which encodes in *reverse* order the sequence in which they can fire. The *unordered* set of rules associated with S, $Un(S) = \{R_i, \ldots, R_j\}$, $1 \leq i \leq j \leq n$, is the set of rules without any ordering, less duplicates. The *logical frontier* of S, $Fr(S)$, is the propositional formula implied by S in terms of *leaves* of the inference graph. Usually, a leaf is equivalent to a finding, but in an incomplete knowledge base may represent a hypothesis. The *consequent* of S, $Con(S)$, is the consequent of the first rule R_1 of this sequence.

The above terminology is illustrated by the following example.

Example 2.1

Consider the set of rules and associated inference graph given for figure 1 in the next section. The subgraph consisting of (non-leaf) nodes $\{h_0, h_1, h_2, h_3, h_4\}$ in the above graph admits the ordering $\langle 4, 2, 1, 3, 6 \rangle$ (there are others) with the associated ordered set $S = \langle R_4, R_2, R_1, R_3, R_6 \rangle$, and

$$Un(S) = \{R_4, R_2, R_1, R_3, R_6\}$$
$$Fr(S) = ([1 : f_1, f_2] \ \& \ f_3) \ \& \ (f_9 \ \& \ [1 : f_5, f_6])$$
$$Con(S) = h_4$$

In illustrating the method, it is useful to introduce a simple form of term rewriting which shows the construction of an ordered set in a step-by-step manner. We will say that h_i in a propositional formula P is rewritten by rule $R = X_1$ & ... & $X_n \rightarrow h_i$ if a single term h_i in P is replaced by X_1 & ... & X_n.

We are now in a position to present our technique.

3 Rule interactions: the S/G-relation

The problem introduced by adding a new rule to a knowledge base is that the knowledge engineer must check that all possible inference chains involving the new rule "make sense". This imprecise notion of "making sense" is related to the fact that the measure of *correctness* of a consultation system is how closely it matches the domain expert's conclusions. Correctness is defined in terms of matches, false positives, and false negatives, rather than in a formalizable notion such as logical consequence. This is one of the essential differences between a conventional database and a knowledge base associated with a consultation system. As such, the approach discussed here attempts to point out interactions between ordered sets of rules (chains) that make "questionable" sense. We assume that a pairwise comparison has been made between the extant rules and the new rule in the manner of [13] to identify obvious lexical errors, logical inconsistencies, redundancies, and other pairwise relations. Here we consider additional more subtle problems that the addition of a new rule may present.

The basic idea is that the addition of a new rule can create a new ordered set which has a logical frontier which either logically implies a logical frontier of an existing ordered set, or is implied by a logical frontier of such a set.

Definition 3.1 Let R be a rule to be added to a knowledge base with associated inference graph G, such that $N = \{n \mid n \text{ is a node in } G \text{ associated with } R\}$. Let $\mathcal{OS} = \{S \mid S \text{ is an ordered set such that (1) } conseq(S) = h, \text{ (2) } \forall n \in N,\ n \leq h, \text{ and (3) if } \exists h' \in G \text{ such that } \forall n \in N,\ n \leq h', \text{ then } h \leq h'\}$. In words, $\mathcal{OS}$ is the set of all ordered sets of rules whose consequent is the unique node in G which minimally subsumes all subgraphs in G for which some node in R is a member. If for any $S_1, S_2 \in \mathcal{OS}$, $Fr(S_1) \supset Fr(S_2)$, then S_1 *specializes* S_2, symmetrically that S_2 *generalizes* S_1, and that R induces an instance of the *specialization generalization (S/G)-relation* between $Un(S_1)$ and $Un(S_2)$, denoted as $Un(S_1) \Longleftrightarrow Un(S_2)$.

We adopt the convention that the right-hand side of the S/G-relation refers to the generalizing set. The dropping of the ordering is important, as we want the S/G-relation to provide information about the rules involved, not the order in which they were used to establish a consequent. We acknowledge that the S/G-relation is in in some sense "obvious", but argue that properly used, the relation can help identify situations that can lead to difficult to maintain and even "incorrect" knowledge bases. Conceptually, adding a rule that induces a new ordered set that generalizes an existing ordered set may not, in fact, be introducing new information. Conversely, adding a rule that induces a new ordered set that specializes an existing ordered set may suggest modifying or deleting the existing ordered set. However, neither action can be categorical because of the nature of "correctness" of a knowledge base associated with a consultation system.

We illustrate the method with the following examples.

Example 3.2

Consider adding the rule $R_7 : f_3 \rightarrow h_2$, indicated graphically by dotted lines, to the following set of rules:

$R_1 : [1 : f_1, f_2] \rightarrow h_1$
$R_2 : h_1 \ \& \ f_3 \rightarrow h_2$
$R_3 : h_0 \ \& \ [1 : f_5, f_6] \rightarrow h_3$
$R_4 : h_2 \ \& \ h_3 \rightarrow h_4$
$R_5 : h_3 \ \& \ f_7 \rightarrow h_5$
$R_6 : f_9 \rightarrow h_0$

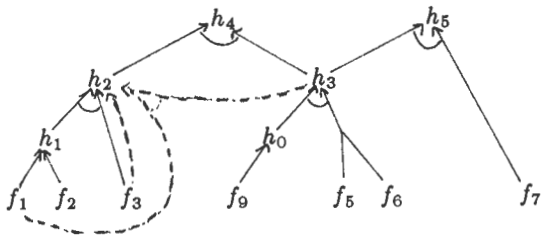


Figure 1 Inference graph for example 3.2

We will ignore for the moment $R_8 : f_1 \ \& \ h_3 \rightarrow h_2$, even thought it is shown in the graph. Thus, we are to rewrite h_2, the unique node which minimally subsumes all subgraphs for which some node in the new rule is a member. Using R_7, h_2 rewrites immediately to f_3. Alternatively, using the sequence of rewrite steps

h_2 by $R_2 \rightarrow h_1 \ \& \ f_3$
h_1 by $R_1 \rightarrow [1 : f_1, f_2] \ \& \ f_3$

we have $[1 : f_1, f_2] \ \& \ f_3 \supset f_3$, $Fr(\langle R_2, R_1 \rangle) \supset Fr(R_7)$, and thus $\{R_2, R_1\} \Longleftrightarrow \{R_7\}$.

Now consider adding rule $R_8 : f_1 \ \& \ h_3 \rightarrow h_2$, which is also shown in dotted lines. A quick check of the inference graph shows that this rule suspiciously appears to introduce dependencies across subgraphs, and we might expect it to induce an instance of the S/G-relation. However, it is not actually obvious what effect the addition of this rule has on the extant set of rules. Rewriting, we get

h_4 by $R_4 \rightarrow h_2 \ \& \ h_3$
h_2 by $R_2 \rightarrow h_1 \ \& \ f_3 \ \& \ h_3$
h_1 by $R_1 \rightarrow [1 : f_1, f_2] \ \& \ f_3 \ \& \ h_3$
h_3 by $R_3 \rightarrow [1 : f_1, f_2] \ \& \ f_3 \ \& \ h_0 \ \& \ [1 : f_5, f_6]$
h_0 by $R_6 \rightarrow [1 : f_1, f_2] \ \& \ f_3 \ \& \ f_9 \ \& \ [1 : f_5, f_6]$

Thus, letting $S_1 = \langle R_4, R_2, R_1, R_3, R_6 \rangle$, we have

$$Fr(S_1) = [1 : f_1, f_2] \ \& \ f_3 \ \& \ f_9 \ \& \ [1 : f_5, f_6].$$

Before the addition of R_8, S_1 is the only ordered set rooted at h_4, that is, the only way to rewrite h_4. The addition of R_8 induces a new set rooted at h_4 however. Rewriting, where R_T stands for contraction by tautological identity,

h_4 by $R_4 \rightarrow h_2 \ \& \ h_3$
h_2 by $R_8 \rightarrow f_1 \ \& \ h_3 \ \& \ h_3$
$R_T \rightarrow f_1 \ \& \ h_3$
h_3 by $R_3 \rightarrow f_1 \ \& \ h_0 \ \& \ [1 : f_5, f_6]$
h_0 by $R_6 \rightarrow f_1 \ \& \ f_9 \ \& \ [1 : f_5, f_6]$

Since, letting $S_2 = \langle R_4, R_8, R_3, R_6 \rangle$, we have $Fr(S_1) \not\supset Fr(S_2)$ and $Fr(S_2) \not\supset Fr(S_1)$, the S/G-relation is not present between these two sets. Note that a slight change of R_8 to $R_8' : f_3 \ \& \ h_3 \rightarrow h_2$, causes an interaction, since $Fr(\langle R_4, R_8', R_3, R_6 \rangle) = f_3 \ \& \ f_9 \ \& \ [1 : f_5, f_6]$, and thus, $Fr(S_1) \supset Fr(\langle R_4, R_8', R_3, R_6 \rangle)$.

Notice that in this example the sets $\{R_4, R_8', R_3, R_6\}$ and $\{R_4, R_2, R_1, R_3, R_6\}$ have rules which are not in their intersection, namely $\{R_8', R_2, R_1\}$. Identifying such rules narrows the scope of the interaction. We will refer to such an occurence as an instance of the S/G-relation with an associated *subset component*, denoted by $\overset{s}{\longleftrightarrow}$. Thus, in the above example,

$$\{R_4, R_2, R_1, R_3, R_6\} \Longleftrightarrow \{R_4, R_8', R_3, R_6\} \quad \text{with}$$
$$\{R_2, R_1\} \overset{s}{\longleftrightarrow} \{R_8'\}$$

It should be noted that for two sets that induce an instance of the S/G-relation, it is *not* always the case that

(1) independently, elements not in the intersection of the sets induce an instance of the S/G-relation, (here R_8' and R_2 are not related), or (2) the intersection is nonempty. Point (1) shows that the approach of pairwise comparison of rules does not identify all instances of rule interaction, and point (2) shows that the subset relation provides additional information.

The next example trivially illustrates this last point.

Example 3.3

Consider adding $R_9 : [1 : f_2, f_7] \rightarrow h_5$, which is shown in figure 2 in dotted lines (as is the addition R_{10} for the next example) to the example set of rules.

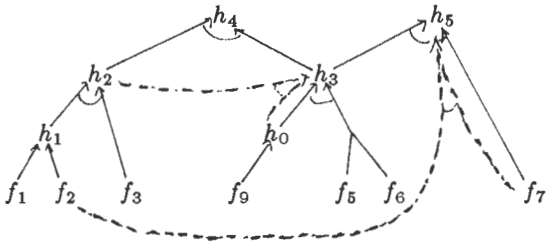


Figure 2 Inference graph for example 3.3

Then h_5 immediately rewrites to $[1 : f_2, f_7]$ by R_9, and by an alternate sequence to

h_5 by $R_5 \rightarrow h_3$ & f_7
h_3 by $R_3 \rightarrow h_0$ & $[1 : f_5, f_6]$ & f_7
h_0 by $R_6 \rightarrow f_9$ & $[1 : f_5, f_6]$ & f_7.

Therefore f_9 & $[1 : f_5, f_6]$ & $f_7 \supset [1 : f_2, f_7]$, and we have $Fr(\langle R_5, R_3, R_6\rangle) \supset Fr(\langle R_9\rangle)$, and thus $\{R_5, R_3, R_6\} \Longleftrightarrow \{R_9\}$, but the intersection is empty. Thus the subset observation does provide additional information.

We will have suggestions on how this information can effectively be used in the next section. It should be pointed out that pairwise comparison of rules would not have identified the interactions between the ordered sets of rules in the above examples. We illustrate the method again in the final example.

Example 3.4

Consider the initial set of rules with addition of rule 8, and subsequently the addition of the rule $R_{10} : h_2$ & $h_0 \rightarrow h_3$, illustrated in figure 2. Rewriting,

h_4 by $R_4 \rightarrow h_2$ & h_3
h_2 by $R_2 \rightarrow h_1$ & f_3 & h_3
h_1 by $R_1 \rightarrow [1 : f_1, f_2]$ & f_3 & h_3
h_3 by $R_{10} \rightarrow [1 : f_1, f_2]$ & f_3 & h_2 & h_0
h_2 by $R_2 \rightarrow [1 : f_1, f_2]$ & f_3 & h_1 & f_3 & h_0
$R_T \rightarrow [1 : f_1, f_2]$ & f_3 & h_1 & h_0
h_1 by $R_1 \rightarrow [1 : f_1, f_2]$ & f_3 & $[1 : f_1, f2]$ & h_0
$R_T \rightarrow [1 : f_1, f_2]$ & f_3 & h_0
h_0 by $R_6 \rightarrow [1 : f_1, f_2]$ & f_3 & f_9

Eliminating duplicates right to left, we have the ordered set $\langle R_4, R_2, R_1, R_{10}, R_6\rangle$. Rewriting h_4 by another sequence, we have

h_4 by $R_4 \rightarrow h_2$ & h_3
h_2 by $R_2 \rightarrow h_1$ & f_3 & h_3
h_1 by $R_1 \rightarrow [1 : f_1, f_2]$ & f_3 & h_3
h_3 by $R_3 \rightarrow [1 : f_1, f_2]$ & f_3 & h_0 & $[1 : f_5, f_6]$
h_0 by $R_6 \rightarrow [1 : f_1, f_2]$ & f_3 & f_9 & $[1 : f_5, f_6]$

Since for this sequence, $[1 : f_1, f_2]$ & f_3 & f_9 & $[1 : f_5, f_6] \supset [1 : f_1, f_2]$ & f_3 & f_9, this induces

$$\{R_4, R_2, R_1, R_3, R_6\} \Longleftrightarrow \{R_4, R_2, R_1, R_{10}, R_6\} \quad \text{with}$$
$$\{R_3\} \overset{\delta}{\longleftrightarrow} \{R_{10}\}$$

A subset relation between singleton sets is very strong evidence that one of $\{R_3\}$ and $\{R_{10}\}$ should be changed.

4 Making use of rule interaction information

We use the information described in the last section as follows. After a new rule or group of rules has been added to the knowledge base, the system returns all instances of the S/G and subset relations. We then use this list of instances of the S/G-relation to check for *patterns* that indicate that the "correctness" of the knowledge base has been called into question. Patterns that appear to be domain independent include the following:

(1) *Specialize-low.* $S_{new} \Longleftrightarrow S$, and yet S_{new} propagates a lower confidence factor. We argue that such an interaction probably does not preserve the "correctness" of the knowledge base. Let X range over some set of objects, and I be an instance of $X' \subseteq X$. A set of rules S_{new} that correctly classifies the instance I of X' also classifies I as a member of X. To make a subclassification, S_{new} requires information that refines, or specializes the set S. Since we are using confidence factors to measure degree of belief in class membership, the confidence factor of S_{new} should be set at least as high as any generalization.

(2) *Generalize-high*: $S \Longleftrightarrow S_{new}$, yet S_{new} propagates a higher confidence factor. This is the dual of the previous pattern.

(3) *Many-specializations*: If S_{new} generalizes many existing sets, the user could consider rewriting some specializations of S_{new} to an intermediate hypothesis. This will improve the structure of the knowledge base without changing its classification performance.

(4) *Many-generalizations*: If S_{new} specializes many existing sets, this may be an indication of a weakness in identifying the consequent hypothesis. The user could consider discarding or consolidating several general rules into a smaller number of more specific rules. Such a decision could increase the number of false negative classifications,

and is thus dependent on the domain expert's judgement that more specific rules will preserve the integrity of the knowledge base with respect to the consequent of S_{new}.

We restate again that due to the imprecise nature of "correctness" of a consultation knowledge base, exceptions are always possible, and it is incorrect for a system to categorically effect changes. In the domain of fault diagnosis, for example, an exception to the specialize-low pattern described above is the following: the apparent co-occurrence of two fault hypotheses may constitute negative evidence for the presence of either fault alone. In this case a rule may propagate a higher negative confidence than its positive weighted generalization. In general, however, the above patterns should be checked carefully.

5 An implementation note

As mentioned, the method presented here is dependent on identifying the logical frontiers of ordered sets of rules in subgraphs of an inference graph, which can be viewed as computing a type of formula in disjunctive normal form. Since the complexity of this computation is NP-complete, it might seem that no practical tool could based on such a method. However, there are practical considerations related to the nature of inference graphs and the inclusion of heuristics which make this method viable in practice. In a paper of this scope, we cannot "prove" this statement, as it would require a detailed analysis of the structure of knowledge bases for consultation systems. We can, however, give a short discussion of these considerations along with an example heuristic as an "argument" for the viability of the approach. Reference [6] contains an extended example and detailed treatment of this issue.

Inference graphs have an important property related to their use that distinguishes them from arbitrary graphs: they attempt to classify a set of observations by partitioning the hypothesis space. On the basis of this property, it is reasonable to make the following two assumptions.

1. The partition is *approximately* uniform, i.e., no single final hypothesis is associated with a disproportionately large subsection of the inference graph.
2. The partition of the hypothesis space is sufficiently fine-grained so that the inference graph does not have the structure of a "thin tree". This will be the case for almost all non-trivial knowledge bases.

These assumptions have the effect of constraining the computation of the set of all ordered sets for a specific inference graph G. If the number of rules associated with G is N, and the number of final hypotheses is k, the assumptions imply that, within a small constant factor c for overlapping, the number of rules associated with any final hypothesis h is $(cN)/k$. Within these $(cN)/k$ rules, there will be only m rules which establish the same consequent. It is the parameter m which results in alternate ordered sets for h, and the worst-case complexity of $O(2^m)$. For example, in example 3.2 with the addition or R_7 (but not R_8), there are six rules which are associated with h_4 and two which establish the same consequent, h_2. There are four ordered sets for h_4, $\langle R_4, R_2, R_1, R_3, R_6\rangle$, $\langle R_4, R_7, R_3, R_6\rangle$, and two permutation instances of these sets. The final hypothesis h_5 has only one ordered set associated with it.

A pratical implementation would have to identify permutation instances. In a domain that does not fit this definition of uniformity, the method could be used by introducing intermediate hypotheses to further partition the hypothesis space. Thus, our system also identifies rules that cause the hypothesis space to be "unbalanced", which in and of itself may be useful information.

An example of a linear-time heuristic is the following. Let $\{X_1, \ldots, X_n\}$ be the set of finding and hypothesis *symbols* in an ordered set of rules S_1, and $\{Y_1, \ldots, Y_m\}$ the symbols in S_2. Then an instance of the S/G-relation between S_1 and S_2 is less likely to occur if $\{X_1, \ldots, X_n\} \not\subseteq \{Y_1, \ldots, Y_m\}$ and $\{Y_1, \ldots, Y_m\} \not\subseteq \{X_1, \ldots, X_n\}$.

Our implementation (in progress) of the method maintains a queue of "events" that it believes that the user should be aware of. Examples of such events include an instance of the S/G relation, and something like the following: a change to the structure of the graph that causes some findings, intermediate hypotheses, and final hypotheses to exchange roles. Deleting a rule that has a consequent intermediate hypothesis may appear to create a finding if the number of incoming arcs of the hypothesis is reduced to zero, for example. Basically, an event is anything that we felt that a knowledge engineer would want to be warned of while constructing a knowledge base. After each addition of a rule or after a session, the system asks the user if he wishes to view this queue.

6 Discussion and future work

Despite claims made by various vendors of knowledge-based software tools, it is the authors' opinion that no domain independent tool currently exists (or is likely to be developed very soon) that replaces either the domain expert or the knowledge engineer. We have instead addressed the problem of facilitating the efficient transfer of knowledge, by identifying rules whose addition can participate in interactions between sets of rules. We have the knowledge engineer mediating on issues of knowledge representation, which are neither within the domain expert's range of competence or interest [2,3,4], and the domain expert as the final evaluator of the "correctness" of the consequences of the knowledge transfer. Reboh [7] sees the knowledge engineering task as separable into subtasks of knowledge base design and coding. We have found that the method helps in a small way to automate a part of the function of the coder, and by drawing the domain expert

more actively into the process, provide him with a better understanding of the issues faced by the knowledge base designer. The increased involvement of the domain expert may in addtion have motivational value, especially if it can be achieved without requiring him to learn the details of knowledge base coding.

In addtion to work on an efficient implemtation of this method, we are currently extending this work in three areas:

(1) The identification and interpretation of specific knowledge engineering patterns related to the fault diagnosis domain.

(2) The present implementation does not incorporate taxonomic relationships. Consider rule R_1 with consequent h_1, and rule R_2 with consequent h_2 where h_1 is related to h_2 through a taxonomic hierarchy. The taxonomic relation is a kind of implied inference link that propagates confidence from subclasses up to a superclass. We plan to investigate how to include taxonomic relationships in the method.

(3) Early in the development of a model a database of cases is not available, and we feel it is important that the system not depend on this information. However, we plan to include the ability to perform empirical performance analysis of a model over such a database, mostly for the purpose of maintaining and refining an extant knowledge base [5].

Acknowledgement: We would like to thank Bonnie Webber and Tim Finin for their helpful comments.

7 References

[1] Clancey, W.J., Heuristic Classification, *Artificial Intelligence* 27:3, (1985), 289-350.

[2] Davis,R., Interactive Transfer of Expertise: Acquisition of New Inference Rules, *Artificial Intelligence* 12 (1979), 121-157.

[3] Drastal, G. and Kulikowski, C., Knowledge-Based Acquisition of Rules for Medical Diagnosis, *J. of Medical Systems*, Vol. 6:5 (1982), 433-445.

[4] Drastal, G., Experiments with Rule Writer, A Tool for Building Expert Systems, Proc. 17^{th} Hawaii International Conference on System Sciences 1984, Honolulu, HI.

[5] Politakis, P. and Weiss, S., Using Empirical Analysis to Refine Expert System Knowledge Bases, *Artificial Intelligence* 22 (1984), 23-48.

[6] Raatz, S. and Drastal, G., Rule Interaction in Expert System Knowledge Bases, tech. rep., University of Pennsylvania, 1986.

[7] Reboh, R., Knowledge Engineering Techniques and Tools in the Prospector Environment, TR 243 (June 1981), SRI International, Menlo Park, CA.

[8] Reboh, R., Using a Matcher to Make an Expert Consultation System Behave Intelligently, AAAI 1980, 231-234.

[9] Suwa, M., Scott, A., and Shortliffe, E., An Approach to Verifying Completeness and Consistency in a Rule Based Expert System, *AI Magazine*, Fall 1982, 16-21.

[10] van Melle, W., A Domain-independent Production-rule System for Consultation Programs, *IJCAI* 79, 923-925.

[11] Weiss, S. and Kulikowski, C., *EXPERT*: A System for Developing Consultation Models, *IJCAI* 79, 942-947.

[12] Weiss, S., Tools for Knowledge Engineering, Expert System Workshop, August 1980, San Diego.

[13] Nguyen, T. et al, Checking an Expert System Knowledge Base Consistency and Completeness, *IJCAI* 85, 375-378.

Towards user specific explanations from expert systems

Peter van Beek and Robin Cohen

Department of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1

Abstract

This paper presents a model for generating user specific explanations, based on tracking the user's stated, intended and overall domain goals. The work may be seen as extending the research of [Joshi et al. 84] to include different explanations for different users, and to be domain independent. The model works by constructing possible user domain plans and comparing them to system-generated plans, to provide the best alternative to a user's query. As a result, the model is particularly useful for detecting and correcting user's misconceptions. Finally, an extended environment for this model is studied - suggesting expert system situations where user models are most appropriate and strategies for variation of both form and content for specific users.

1. Introduction

> Man may smile and smile but he is not an investigating animal. He loves the obvious. He shrinks from explanations. Yet I will go on with mine.
>
> -- Joseph Conrad --

> The positivist separation of logic and pragmatics meant that for many years pragmatics was the Cinderella of language, forced to stay home and do the dirty work while sisters syntax and semantics received all the attention.
>
> -- Alan Garfunkel --

Explanation capabilities are an important area for research in computational linguistics. They are of theoretical interest as a sub-problem in the larger area of generation of coherent natural language text, but they are also of practical importance for computer systems in language intensive fields such as computer aided instruction and expert systems. The communication needs of these types of programs have increased the necessity for sophisticated explanation capabilities.

Users of expert systems judge a system's explanations by the criteria of relevance, utility in practice, informativeness, and speaking to the question. Thus, there is a pragmatic component to explanation: it requires reference to the person to whom the explanation is directed. An explanation should depend on the beliefs, goals, knowledge, and assumptions of a user on a particular occasion. The pragmatic processing needed to produce improved, user specific explanations requires a model of the user. As a step in this direction we show how a model of the user that incorporates some information on the background, goals, and plans of the user can be used to produce better explanations. The type of domain independent reasoning that the explanation facility must go through to make use of the user model will be detailed.

Previous research has investigated user modelling in the context of interpreting an utterance correctly and of generating appropriate explanations. In MYCIN [Shortliffe82] and Teiresias [Davis82], both medical diagnosis expert systems, the user can specify his/her level of expertise when requesting an explanation. While this approach allows the explanation to be tailored to the level of knowledge of the user, it does not address why an explanation was requested or the other domain goals of the user. An alternative approach is to gather information about the user's goals, assumptions, and knowledge from the user/system dialogue (eg. [Allen82], [Carberry83], [Joshi et al. 84], [Litman and Allen84], [McKeown et al. 85], [Pollack84]). This approach is particularly fruitful in the expert system environment where the user and the system work cooperatively towards common goals through the dialogue and the user's utterances may be viewed as actions in plans for achieving those goals. The present work draws on this latter approach. We propose a user model based on the goals and plans of the user derived from the discourse together with his/her more static higher domain goals and plans. This differs from McKeown's approach in that we are interested in identifying possible user misconceptions, avoiding misleading responses, and comparing and suggesting possible alternative plans.

2. Generating Better Explanations

The current work is an extension of the approach in [Joshi et al. 84] (hereafter referred to as "Joshi's" for simplicity). Joshi focuses on characterizing the types of informing behaviour usually expected of an expert. In particular, an expert system must modify the planned response if there is reason to believe that the response may mislead the user. To identify these cases he makes use of the user's stated goal (the goal directly achieved by using the information requested) and his/her intended goal (the goal underlying the stated goal of a request). In addition to the direct, correct answer, a user expects to be informed if, for example, the stated goal does not achieve the intended goal or if there is a better way to achieve the intended goal (refer to Figure 1). If the expert system does not supply this additional information, P, it risks misleading the user. The user may interpret the expert system's silence regarding P as implying that P is not true.

Joshi's approach can provide an additional or modified response in order to block inappropriate default reasoning on the part of the user. These modified responses also have intrinsic worth in that they provide additional useful information to the user. However, his approach can compute the above only as it relates to the stated and intended goal of the immediate question. By adding the user's higher domain goals/plans to the user model we can identify additional cases where a response must be modified in order to prevent false inferencing. Our approach to reasoning about goals and plans is domain independent vs. Joshi's approach of precompiling

Scenario:

The stated goal of the user is not being in the course and the intended goal of the user is to avoid failing the course.

User:

Can I drop numerical analysis?

System:

a) Yes, however you will still fail the course since your mark will be recorded as withdrawal while failing.

b) Yes, but a better way is to take an incomplete to have more time to perform the work.

Figure 1 Example from [Joshi et al.84]

"the conditions into a case analysis similar to a discrimination net". In Joshi's approach alternatives are explicitly enumerated and comparing alternatives is done by a primitive that is not user specific. The introduction of additional, higher domain goals (eg. in the course domain: the goal of getting a degree with certain course requirements) provides some insights into generating and comparing alternatives with regards to a specific user.

As test cases for our model of explanation the application areas of requests for information from a student advisor system and an educational diagnosis expert system were chosen, and will be illustrated in this paper. The requests will be about the possibility (eg., "Can I drop numerical analysis?") or advisability (eg., "Is the McLeod Phonics Test appropriate?") of an action. An appropriate response will include an indication of the possibility and advisability of the action, better alternatives and consequences of the actions if appropriate, and an answer to the implicit question of why the result should be believed by the user.

The following section will show when and why different explanations may be required depending on the immediate and higher domain goals of a specific user. A subsequent section will detail the procedures for generating these different explanations.

3. User Specific Explanations

An explanation can be tailored to a specific user at a number of different levels, including by content [McKeown et al. 85], discourse strategy used ([Mann and Thompson83], [McKeown et al. 85]), level of detail [Paris85], and vocabulary and surface structure [McKeown and Derr84]. We focus on how the content of an explanation should change as a function of the goals of a discourse situation and on the higher domain goals of the user.

We have identified several ways in which an explanation can be user specific by providing additional relevant information that is aimed at blocking inappropriate default reasoning or at simply being cooperative. In planning a response, the system should ensure that the current goals, as expressed in the user's queries, are consistent with the user's higher domain goals. If a user presupposes that an action has a positive effect, the system must ensure that this is the case. Conversely, if an action is known to have a negative effect, the system must ensure that the negative consequences are not greater than assumed by the user.

For example, in the course domain the user may have as a current goal to enroll in a course and a higher domain goal of achieving a degree. The system assumes that the user believes the current goal will help achieve the higher domain goal. If the relationship between the current goal and the higher domain goal does not hold, the system must inform the user. On the other hand, in the course domain the user may have as a current goal to drop a course which has a negative effect towards the domain goal. If the system knows that this has an additional negative effect, such as the course is a co-requisite to another of the user's courses, the system must inform the user.

The expert system can be additionally cooperative by suggesting better alternatives if they exist. Furthermore, both the definitions of better and possible alternatives should be relative to a particular user.

Scenario:

The child appears to have a language difficulty. He doesn't speak much and when he does his sentences are short and choppy with connecting words, prepositions, and articles usually missing. Testing is aimed at getting a measure of intelligence to determine whether this disability reflects mental retardation or just slow development.

User:

Is the Neale Analysis of Reading an appropriate test?

System:

a) The Neale is a useful test of reading comprehension and reading within context, but it is not an I.Q. test. The PPVT (Peabody Picture Vocabulary Test) would be useful in this situation.

b) While the Neale will be a useful test if the child's difficulties prove to be the result of slow development, the general level of performance indicates that testing should be aimed at getting a measure of intelligence to determine whether this disability reflects slow development or mental retardation. The PPVT is a good test to give in this situation.

Figure 2 Example from educational diagnosis domain

In each of the cases in figure 2, the explanation points out that a goal of the user is inappropriate, provides background information aimed at relieving the user of his/her misconception, and suggests a better alternative. Response (a) informs the user that the stated goal (administering the Neale test) does not achieve his/her intended goal (administering an I.Q. test). Response (b) addresses not the relationship between the plan of the user and its intended goal, but instead addresses the inappropriateness of the intended goal as it relates to the user's higher domain goal of planning a remedial program for the child under consideration. The user's goals are necessary for computing a correct and best response to the user's queries. An explanation that did not address why the user asked a question and his/her overall domain goals would be unsatisfactory to the user and may even fail to be explanatory.

4. Procedures for Utilizing the User Model

Our model requires a database of domain dependent plans and goals. A plan may contain subgoals, actions, and constraints (cf. [Sacerdoti77], [Litman and Allen84]) The plans are hierarchical since subgoals may themselves have associated plans. We assume that the goals of the user in the immediate discourse are available, perhaps by methods specified in ([Allen82], [Pollack84], [McKeown et al. 85]).

It is proposed that the model of a user contain, in addition to the user's immediate discourse goals, his/her higher domain goals and plans; these plans specifying how the higher domain goals will be accomplished. The plans may be neither completely specified nor instantiated, either because the user has not completely defined his/her plan or because the system has an incomplete knowledge of that definition. To give this some concreteness, consider an example from the student advisor domain. A higher domain goal of the student is very likely to be the attainment of a degree. Initially, the plan to obtain that goal will be an incomplete, uninstantiated plan that contains only the constraints imposed by the student's university. Parts of the plan may be corrected (eg. the student changes his/her major) or expanded and instantiated (eg. the student completes and enrolls in courses) based on the actions of the student. New goals and plans will be added to the model (eg. the student's preferences or intentions) as they are derived from the discourse.

To recapitulate, the user model contains the stated goal (s-goal) and the intended goal (i-goal) upon which the current discourse is focused, and some higher domain goals. A representative list of things which must be checked for to identify possible misconceptions on the part of the user and thus to provide the best response, are the following:

1. The i-goal may already be true.
2. The i-goal may not be possible to achieve.
3. Several relationships that can hold between a stated goal (s-goal) and an intended goal (i-goal) can be enumerated (cf. [Joshi et al. 84]):
   - 3.1 The s-goal may be the same as the i-goal.
   - 3.2 The s-goal may be a subgoal that addresses only part of the i-goal. For example, the user's s-goal may be to enroll in a natural language course while his/her i-goal may be to concentrate on ai.
   - 3.3 The s-goal may be a pre-condition of the i-goal. For example, the s-goal may be to get read/write access to a file, while his/her i-goal may be to alter it.
   - 3.4 The i-goal may be more specific than the s-goal. For example, the s-goal may be to know how to send files to someone on another machine, while his/her i-goal is just to send a file to a particular machine, which may allow for a specialized procedure.

The cases enumerated above also describe the relationships that hold true between the stated goal and higher domain goals, and between the intended goal and higher domain goals. The user may falsely believe that a certain relationship holds between his goals; it is then up to the system to correct the misconception. Given our recasting of the knowledge representation in terms of goals and plans for achieving those goals, a natural method of computing whether these relationships hold presents itself. It also allows us to identify and present to the user knowledge that may make the misconception clear. Thus, (3.1) corresponds to checking for equality, (3.2) involves checking whether the s-goal could be a possible step in the plan for the i-goal, (3.3) whether the s-goal is a pre-condition in the plan for the i-goal, and (3.4) whether the i-goal is among the (possibly) multiple plans available for achieving the s-goal.

If the relationship does not hold, the user should be informed. In addition, the system should ensure that, even if the relationship does hold, the plan is executable. Here it is possible to provide additional unrequested information necessary to achieve the goal (cf. [Allen82]). But the system must also inform the user of (1) any pre-conditions that can't possibly become true, given the state of the world, and thus preclude the possibility of achieving the goal (figure (1a)), and (2) whether the usefulness of the s-goal is dependent on the result of first achieving an all together different goal (figure (2b)).

Scenario:

University regulations state that at least five courses taken by the student must be numbered greater than or equal to 600, at least two > 200, and at least one > 600 in at least four of six possible areas. The student has already met his/her 600 level requirement and the course asked about is in an area already covered.

User:

Can I enroll in CS 677?

System:

Yes, but it will not get you further towards your degree since you have already met your 600 level requirement and have already taken a course in that area. Some useful courses would be CS 775 and CS 634.

Figure 3 Example from student advisor domain

Until now we have discussed a model for generating better, user specific explanations. A test version of this model has actually been implemented, in Prolog. We discuss below how this system works.

The system considers the background of the user (in the course domain: the courses taken), the background of the domain (in the course domain: what courses are offered) and a query from the user (eg. "Should I take cs492?"), and ensures that the goal of the query is consistent with the attainment of the overall goal.

In the hypothetical situation described in figure 3, the relationship between the user's s-goal of enrolling in a course and the higher domain goal of achieving a degree does exist but several pre-conditions fail. That is, given the background of the user the goal of the query to add cs677 will not help achieve the overall domain goal. The failed pre-conditions, together with the knowledge that taking courses is expected to have a positive impact, are used to form the first sentence of the system's response.

To suggest better alternatives, the system goes into a planning stage. There is stored in the system a general plan for accomplishing the higher domain goal of the user. This plan is necessarily incomplete and is used by the system to track the user by instantiating the plan according to the user's particular case. The system considers alternative plans to achieve the goal at the level of the user's query. For this particular example, the system discovers other courses that the user can add which will satisfy the higher goal.

goal	sub-goals (+ constraints)
achieve_degree	<- requirement1,requirement2,requirement3
requirement1	<- courses_taken, course(Course_No,Area) + Course_No > 600 and sum of courses taken > 4
requirement2	<- courses_taken, course(Course_No,Area) + Course_No > 700 and sum of courses taken > 1
requirement3	<- courses_taken, course(Course_No,Area) + sum of unique areas > 3

Figure 4 Simplified domain plan for course domain.

To actually generate alternatives, a module of the implemented system is a horn clause theorem prover built on top of Waterloo Unix Prolog. The theorem prover generates possible alternative plans by performing deduction on the goal at the level of the user's query. That is, the goal is "proven" given the "actions" (eg. enroll in a course) and the "constraints" (eg. prerequisites of the course were taken) of the domain. In the example of figure 3, the expert system has the following horn clauses in its knowledge base:

course (cs755, systems)
course (cs634, numerical)

Applying the theorem prover to requirement2 (see figure 4) will instantiate the variables *Course_No* and *Area* if such a course and area exist which satisfies the constraints. In this example, cs755 is such a course and enrolling in the course would be a valid step or action in a plan for achieving a degree.

A history of the deduction is kept which can be consulted to explain why an attempted "prove" failed. The history tree is also useful in determining whether a user's plan will help achieve a goal. A search of the successful branches of the history tree will determine whether the user's plan is among the possible successful plans.

Note that the system will generate alternative plans even if the user's query is a step in the domain plan. In this case the challenge is to find a better solution for the user. The (possibly) multiple plans are then potential candidates for presenting to the user. These candidates are then pruned by ranking them according to the heuristic of "which plan would get the user further towards his/her goal". Thus, the better alternatives are the ones that help satisfy multiple goals or multiple conditions. For the example in figure 3, the suggested alternative of cs775 helps satisfy requirement2 and requirement3 and so is preferable.

The system has the ability to further reduce the alternatives displayed to the user. This can be done by employing domain dependent knowledge. For the course domain a rule of the form: "A mandatory course is preferable to a non-mandatory course", may help eliminate presentation of certain options. Another way in which the system can reduce alternatives is to employ previously derived goals of the user such as those that indicate certain preferences or interests. In the course domain, for instance, the user may prefer taking numerical analysis courses. Finally, the system could combine these two methods. For the educational diagnosis domain this involves reference to the case history of the child being diagnosed (model of child) and domain specific knowledge on what makes a test better - such as the rule: "Tests in which the mode of presentation or the mode of response is the same as that in which the child is having difficulty are less appropriate than those in which they differ".

5. An Extended Environment for User Modelling

There are some application areas for expert systems which are particularly well-suited for the employment of user models and to accommodating entire classes of users. One example is the educational diagnosis domain, explored in [McLeod and Jones 85] and [Cohen and Jones 85]. This expert system produces a diagnosis of a student's learning disabilities, with a view to developing a remedial program for the student. A model of the student is then one of the knowledge bases in the system. The intended user of the system is a school psychologist, who is familiar with various educational tests and their predicted diagnoses for particular results from students. This system would ideally be accessible to other kinds of users as well - the classroom teacher, who is familiar with the performance of the student, but perhaps not with the full battery of psychological tests, and the parent, who has additional background information on the child, but is unaware of most test results and educational terminology. This application area thus presents additional challenges for the incorporation of user models. The user's background knowledge of the domain and of the student (the topic) is critical and must be monitored.

One way to deal with the variety of possible users for this expert system is to identify plausible models of the user's knowledge (of the domain, of the student) according to the user's "class". For example, a parent is not assumed to know any facts surrounding test results, but is assumed to know "family history" experiences of learning disabilities. In this way, a starting point for modelling the user's background can be established.

Then, the system should be flexible enough to modify its user model when subsequent dialogue makes clear that the present user model is not quite accurate. The first case is when a user doesn't "know enough". For a user who is a psychologist, the system assumes the user understands correctly why a certain diagnosis results from a certain test situation. Instead, the user may request defense of the system output, signalling a lack of background or disagreement with the system's knowledge[1]. The other extreme "misjudgment" of user model is when the system assumes minimal background of the user, and the user in fact possesses more knowledge - eg. a parent who also happens to be a psychologist.

The question of maintaining and employing a user model for a situation of extended dialogue thus arises. Modelling the domain plans and goals is one critical operation to successful communication from a system. Thus, the model described in this paper is indeed extremely useful. But what is also needed is a kind of process model of what it takes the user to decipher the proposed output from the system. This leads to a consideration of not only the content but the form of the explanation.

Two suggestions are proposed for controlling the form of output. The first is to simply incorporate a model of explanation analysis to guide the generation. The model of [Cohen 83] is in fact well-suited to this application. It prescribes a restricted search for the connections between utterances in analysis of arguments. This limited but reasonable processing can be used as a model of the user's analysis to assure that the connections between the sentences of the system's explanation can be recovered by the user.

[1] The system should also be open to update its possibly incorrect knowledge.

The second suggestion is to make an effort to have the semantic connections between the sentences comprising the explanation understood by the user. Explanations typically omit detail, to be concise and still clear (see [Webber and Joshi 82]). To assure comprehension by the user, some modelling of the user's current beliefs and possible implicit beliefs is useful. Mangement of these belief classes within a user model is a topic for future research.

6. Summary

A model for generating explanations from expert systems is introduced in this paper. In addition, a system built on this model is sketched, using two specific application areas for examples. The same model should be applicable for any domain and explanations are derived for the particular user at hand, critically considering that user's background, so that the output is indeed user specific. Finally, we point out that explanations may be user specific in a variety of ways, and propose that the model of this paper ideally be used in conjunction with methods for tailoring the form of output, and monitoring the user's knowledge of the domain.

7. Acknowledgement

This research was partially supported by NSERC (Natural Sciences and Engineering Research Council of Canada).

8. References

[Allen82] Allen, J.F., "Recognizing Intentions from Natural Language Utterances," *Computational Models of Discourse*, M. Brady, Ed., MIT Press, 1982

[Carberry83] Carberry, S., "Understanding Pragmatically Ill-Formed Input," *Proceedings of AAAI-83*, Washington, DC

[Cohen83] Cohen, R., "A Computational Model for the Analysis of Arguments," University of Toronto Technical Report CSRG-151, Toronto, ONT

[Cohen and Jones85] Cohen, R. and Jones, M., "Incorporating User Models into Expert Systems for Educational Diagnosis," submitted to *Computational Intelligence* special issue on AI approaches to education, November 1985

[Conrad20] Conrad, Joseph, Author's note to *The Secret Agent*, Penguin Books, 1984

[Davis82] Lenat, D.B., and Davis, R., *Knowledge-Based Systems in Artificial Intelligence*, McGraw-Hill, 1982

[Garfunkel81] Garfunkel, Alan, *Forms of Explanation*, Yale University Press, 1981

[Joshi et al. 84a] Joshi, A., Webber, B., and Weischedel, R., "Living up to Expectations: Computing Expert Responses," *Proceedings of AAAI-84*, Austin, TX

[Joshi et al. 84b] Joshi, A., Webber, B., and Weischedel, R., "Preventing False Inferences," *Proceedings of COLING-84*, Stanford, CA

[Litman and Allen84] Litman, D.J., Allen, J.F., "A Plan Recognition Model for Subdialogue in Conversations," University of Rochester Technical Report 141, Rochester, NY

[Mann and Thompson83] Mann, W.C., and Thompson, S.A., "Relational Propositions in Discourse," Information Sciences Institute Report RR-83-115.

[McKeown and Derr84] McKeown, K.R. and Derr, M.A., "Using Focus to Generate Complex and Simple Sentences," *Proceedings of 10th International Conference on Computational Linguistics*, Stanford, CA

[McKeown et al. 85] McKeown, K.R., Wish, M., and Matthews K., "Tailoring Explanations for the User," *Proceedings of IJCAI-85*, Los Angeles, CA

[McLeod and Jones85] McLeod, J. and Jones, M., "The Development of an Expert System for Computer-Guided Diagnosis of Children's Learning Problems: Some Emerging Problems," *Proceedings of International Conference on the Computer as an Aid for Those with Special Needs*, Sheffield, England

[Paris85] Paris, C.L., "Description Strategies for Naive and Expert Users," *Proceedings of ACL-85*, Chicago, IL

[Pollack84] Pollack, Martha E., "Good Answers to Bad Questions: Goal Inference in Expert Advice-Giving," *Proceedings of CSCSI*, London, ONT

[Sacerdoti77] Sacerdoti, E.D., *A Structure for Plans and Behaviour*, Elsevier AI series, New York, 1977

[Shortliffe82] Shortliffe, E.H. and Wallis, J.W., *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, 1984

[Webber and Joshi82] Webber, B. and Joshi, A., "Taking the Initiative in Natural Language Data Base Interactions: Justifying Why," University of Pennsylvania Department of Computer and Information Science Technical Report MS-CIS-82-1, Philadelphia, PA

DIALECT: an expert assistant for information retrieval

Bassano Jean-Claude
Université de Paris-Sud

LRI bât. 490
91405 ORSAY CEDEX FRANCE

Résumé. *L'objectif central du système prototype DIALECT porte sur l'amélioration du fonctionnement et des modalités d'utilisation des "systèmes documentaires". L'exposé est centré sur le problème de la transformation (automatique) de la requête initiale formulée essentiellement en langage naturel.*

Abstract.

The aim of the project, with the experimental system DIALECT, is to improve retrieval effectiveness and to suggest some flexible uses for information retrieval systems. This research involves the integration of artificial intelligence (AI) and information retrieval (IR) techniques.

The particular problem to be solved here by the "expert system" in the environment of information retrieval is the automatic request reformulation.

Key words.
User interfaces, Expert assistant, Natural language processing, Data base management systems and Information retrieval systems, Information retrieval (Application),

1 System overview :

1.1 Information retrieval systems:

Proposal 1. *The quality of the services usually provided by automatic information retrieval systems has been found inadequate for casual users.*

In bibliographic retrieval, stored information files are processed in response to queries submitted by a population of users. Answers are generated to these queries. The aim is to retrieve bibliographic reference, that is, citations to bibliographic items rather than actual data. A particular citation contains pre-specified attributes and coded values. But it also contains natural language texts.

Normalized and effective content identifiers may be extracted from these natural language texts by automatic indexing techniques. But it is not sufficient to characterize each bibliographic item by using a few normalized content terms reflecting the main subject areas of interest.

The user has to choose the terms, to determine wether a term is to be used as a single term or as part of a complete clause... Most operational retrieval systems require search formulations expressed by boolean term combinations: the terms are related by "and", "or", "not". Certain words occurring in the request texts may be too broad to be useful: it is necessary in this case to generate combinations of terms or "term phrase". Alternatively, certain single term may be too narrow: in which case it is necessary to assign complete groups of related terms.

The main difficulty of on-line retrieval environment is the fact that searches must be conducted iteratively in several steps by the user itself. It is necessary to conduct information searches in several operations, by first submitting tentative queries which are successively refined **based on information retrieved during previous search steps** [9, 12, 13, 15, 16].

1.2 The expert assistant DIALECT:

The principle of the system described here is related to some aspects of artificial intelligence: it is capable of generating new facts, still unrecorded in the request, derivable from other facts already known in the "database". Starting from a REQUEST in natural language, throughout a step by step analysis, it works on the limited kernel built up from the **most relevant** documents which have already been found out. At one time, only the most relevant descriptions of document are selected. The methodology consists of forgoing ahead by first tacking and analysing parts of the texts of one or two sample references. The REQUEST is **automatically developped** and transformed in order to retrieve additionnal documents. This dynamic process is repeated over and over until it reaches the point where it becomes entitled to stop. The system progressively provides the users with sentences and references on documents which can be "connected" to the REQUEST.

This Information Retrieval System is built as an application on the D.B.M.S ADABAS. The whole "inferential and intelligent" interface is a text in PL1 of about 5 000 lines. **The improvement of retrieval effectiveness has been observed [5].**

Proposal 2. *The particular problem to be solved here by the "expert system" in the environment of information retrieval is the automatic request reformulation: the replacement of the starting entries by a variety of "equivalent" sentence formulations. This step involves inferencing steps as well as an expansion of the query statements by addition of related terms. The system uses reformulation rules, meta-rules for the control of the whole inferential process and linguistic models as meta-rules or another expert level for the analysis of texts. At the end of the session the request models may be stored in the long-term data base to act as profiles of user's interest. Simultaneously a set of relevant documents is proposed.*

AI techniques are used to represent the knowledge of the expert about "query (re)formulation":
- (i) a rule-based formalism captures specific domain knowledge used to refine the current "request model".

These rules are the "TEXT" reformulation rules and the "CONTEXT (thesaurus expert,...)" reformulation rules.
They are **automatically** built up by the system from the retrieved bibliographic references. The query may be modified by these rules without intervention and relevance judgements from the user...

As an illustration of this process, let us give an example by singling out one particular inference rule from the set of available rules. The first proposition in the basic structure specifies a sub-pattern to be searched for and matched by a REQUEST row: "SYSTEME POUR UNE EXPERIMENTATION". The second proposition specifying sub-pattern to be added as a consequence of successful matching: "SYSTEME PROTOTYPE". The first proposition includes a source object and a target object belonging to the same class as the source object and the target object of the REQUEST row [i.e: "SYSTEME"]. This reformulation rule is constructed from a sentence of an abstract of a previsiously selected bibliographic item. So the "coTEXTE" and the "(coded)CONTEXTE" of the rule must [also] be "partially" matched by the current form of the request. The coTEXTE of the rule is the whole sentence of the abstract [i.e: "ON SE PROPOSE DE DECRIRE UN SYSTEME PROTOTYPE POUR DES EXPERIMENTATIONS EN DOCUMENTATION"]; the CONTEXTE contains the key words "BASES-DE-DONNEES", "LANGAGE-NATUREL", etc. the names of the authors, etc. [i.e.: the others "coded values" of the bibliographic item].

- (ii) (meta-)rules choose and stop the global search strategy. This global search strategy controls the actions of the query reformulation controller by **building** and selecting packets of reformulation rules. The mechanism of the expert query reformulation system will set off some of these candidate reformulation rules...

Within the interface manager of such an "expert system", "rules" control user/system interactions. They incorporate some knowledge about the different types of users and the information to be displayed or acquired.

Proposal 3 (Basic information).

The rules, that make up this controller's knowledge, identify the particular search strategies to activate given appropriate amounts of information. This information is given in the request, in the user and current search models. A retrieval situation may be characterized by features such as the length of the query, the current interest of the system or the user in recall and/or precision, etc... There is about hundred strategical "rules". The "search strategy" rules select the elementary search methods and their parameters [7]. The methods must first retrieve some of the more relevant bibliographical information with necessarily natural language texts. Rules of a linguistic model construct some reformulation rules from these previously selected sentences of texts ... So the first document retrieval steps act as the selecting of the candidate reformulation rules.

The query (re)formulation and the search process are highly interactive: they act as rule selecting and rules setting in the "expert system". The system goes through a cycle: query (re)formulation, retrieval of candidate rules which are explicitely "built" only at this time ... So the basic information provided in the "knowledge base", which gives the inference rules capable of generating new facts from already existing ones, is the [document collection] itself. About three hundred candidate rules are selected and built, about 50 are set and used.

The system is able to gradually perform a thorough analysis of document content and to construct useful knowledge structures in the subject area of the request. It uses some of the previously constructed structures and may suggest new structures which are sometimes incomplete or ambiguous. The syntax-based analysis of the system is able to deal with the multiplicity of ambiguities which arise with this text material. These difficulties are passed, they don't stopped or seriously disturb the overall information retrieval process.

2 Extensions of the conventional retrieval systems and partial examples of working:

If the system contains explicit representation of knowledge about how to retrieve relevant information and how to expand the query, it necessarily uses a linguistic model. This incorporation of linguistic procedures and "query reformulation" knowledge-based expert system in a retrieval setting leads to added benefits in the form of new and more sophisticated services. The following extensions of the "standard" retrieval services are:

2.1 Natural language front-end:

The use of natural language front-end allows the users **to interact with the system using French in their initial information request.**

The request is "L'EVALATION DES SYSTEMES DE DOCUMENTATION". The linguistic analysis generates a formal representation in some intermediate language for each input query formulation. This representation can be seen as a list of "templates" (- triplets -). This table is called in our reqest model: the "PLAN DE RECHERCHE".

PLAN DE RECHERCHE:

EVALUATION	(VZPX)	SYSTEME	(0.80)
EVALUATION	(VZPX)	DOCUMENTATION	(0.60)
SYSTEME	(VZPX)	DOCUMENTATION	(0.70)

It is stored in the short-term memory and not in the database. The formal intermediate query is then in turn transformed into a set of data base commands that can actually be used as an access to the data base (D.B.M.S: ADABAS-ADACIR). This set of commands is compatible with conventional inverted file technology. [see "PROCEDURE DE SELECTION"].

Grammatical features.

Most of the words of the investigated language are found in a lexicon which provides the following information:
- gender (masculine, feminine or neutral)
- number (singular, plural or neutral)
- a normalized form (which is a word without any grammatical feature, such as voice, mode, tense and so on)
- a grammatical class (which is made up of two parts, the first one setting the class and the second one pointing out a specific feature within the class, as in VP, where V stands for verb and P for past participle). There is about 30 grammatical classes.

When a word is not in this dictionary, an inference procedure is used which gives us the required information. This procedure is based on the word endings and further constraints.

So one or more answers are always proposed after the processing of the "grammatical ambiguities".

List of templates built by the syntax-based analysis [see 2.2].

Any row in the "PLAN" may be outlined as [D, (R, C), A] (p) where

- D is the normalized form of the source "semantic" class (- i.e a word or a set of words which have the same grammatical class and are automatically grouped by the system during the inferential process -)

- A is the normalized form of the target semantic class,

- R is a grammatical class specifying whether there exists a relationship assumed by a verb between D and A (- for instance: VX - for active verb -, VI - for infinitive verb -,... -). If there is no such relationship we note VZ.

- C specifies whether there exists a prepositional link between D and A (- for instance: PX, PR, etc. -) or not (- NX, AX, etc. -). It also specifies the grammatical class of A (- NX, AX -). It should be noted that NX stands for substantive, AX for adjective.

- p is the plausibility of the template. This plausibility is determined by the system according to the nature of the syntactic connection between the components [D, (R, C), A] of the template. A plausibility of about 1.0 shows a strong connection, while a plausibility of about 0.5 shows a weak connection.

The "PLAN" is initiated at the user's REQUEST and is processed away step by step: it reflects the "understanding" of the initial query and of some of the sentences which have been selected by the system.

Query-document comparison [stratégie de sélection].

For the query-document comparison, DIALECT provides, in an implicit way, various advanced features such as search decision trees which the system follows through either "phrase (template)" or normalized term-combination searches or both as appropriate. In this combinatorial searching, when some implied AND fails an "hyper-OR" may be done. The operators AND and OR become less and less strict: for AND the presence of most - rather than all - query terms; for OR the presence of some query terms in a document - rather than the presence of one -. The assignement to the previously retrieved information items of complex linguistic structures for further discriminating may also be necessary. The exact search path followed by the system depends on the user input and on the success of the search so far. The system uses control knowledge specified by the designer to select the appropriate parameters of the search strategies and the appropriate strategy for a given situation (- document retrieval strategy = "SELECTION" strategy of the candidate reformulation rules -).

So the basic structure,which originates the building of the rewriting part of the "TEXT reformulation rules", follows from the consolidating of two triplets known as being interdependent. They own to an "overlapping" or to a relationship of dependence and analogy. The REQUEST is developed from basic structures, which actually provide the main part of the information coming from the sentences "matching" the current REQUEST. What matters then is to find out the basic structures with features displaying some similarities with the representation of one of the row of the "PLAN DE RECHERCHE". Applying of the appropriate rule to a row allows additionnal templates to be obtained and "semantic-paradigmatic" classes to be enlarged [see "MEME-CLASSE"].

PLAN DE RECHERCHE:

EVALUATION (VZPX) SYSTEME

CRITERE (VZPX) EVALUATION

DISCUSSION (VZPX) CRITERE

MODELE (VXNX) EVALUATION

AIDE (VZPX) MODELE

MODELE (VZAX) GLOBAL

SYSTEME (VZPX) EVALUATION

SYSTEME (VZAX) DOCUMENTAIRE

EVALUATION (VZAX) AUTOMATIQUE

EVALUATION (VXPX) ACQUISITION

ACQUISITION (VZPX) DIFFUSION

DIFFUSION (VZPX) DOCUMENT

CLASSES SEMANTIQUES:

MEME-CLASSE=EVALUATION DOCUMENTATION RECHERCHE
EFFICACITE PERFORMANCE PRECISION.
MEME-CLASSE=SYSTEME COUT.
MEME-CLASSE=MODELE METHODOLOGIE.
MEME-CLASSE=GLOBAL ANALYTIQUE.

PROCEDURE DE SELECTION:
- - -

- automatical reformulation of the request
"EVALUATION DES SYSTEMES DE DOCUMENTATION"
(5 TH step) -

So the request enables the user to get more and more responses. After a number of steps, the system will reach a point where it is entitled to stop the process. This point is **automatically** evaluated. It may display a list of the available descriptive information. This information marks the relevant and already retrieved documents from the point of view of the system (- CONTEXTE rules: keywords, authors, contries,... -). So the expert assistant may include additional descriptive and specific information on the "background" of the request in natural language.

2.2 Updating of the request model:

The automatic updating of the request model according to the newer and very relevant retrieved information is another extension. We must be capable of extracting specific passages from "texts or abstracts" in the current answer. A rich source of term relation-ships is wanted and must be "built": **it becomes also necessary to utilize language processing capabilities instead of statistical methods [1].**

Structural analysis.

A thorough structural analysis is neither needed nor tractable. The linguistic model only makes a distinction between some basic dependency types : the linguistic apparatus required for effective domain-independent analysis is not (yet ?) at hand. A decision must then be made about the type of linguistics to be implemented here: essentially syntax-driven with elaborate dictionary specifying "syntactic" markers [10] and a progressive learning of "lexical-semantic" links. This progressive learning leads to a gradual improvement of the linguistic assertions in the context of the request [3, 8]. In DIALECT the linguistic analysis is made by a robust "situation-action" rule-based parser (natural language expert): it may be used to handle semi-grammatical utterances [11].

This analysis does not lead to the detailed inferencing and predicting of new facts that become possible when domain-restricted knowledge bases are included in the language analysis system. But it was our goal with the implementation of the system to show that we can automaticaly build rules out of the relevant document abstracts. The rules capture some query domain knowledge and are then used to refine the query model.

2.3 Friendly interfaces:

The generation of adaptative dialogue structures between users and system during the search operation is available. We provide the users with a large degree of flexibility in choosing how to interact with the system. The system may operate under several modes [4]:

(1) A so called "casual user mode (novice or inexperienced user)" provides the user with a fully transparent process which decides on its own on any opportunity to improve the request. The user only submits his query in natural language (- French -) and lets the system search and display the relevant available information as a list of retrieved references and/or certain passages of texts.

(2) The so called "expert assistant (documentalist) or expert user mode" allows the trained user to break more frequently into the process, if he wants to **improve and control its returns.** Such an improvement may consist in redefining some elements of the "semantic classes", adding or removing propositions (template), using underlying models of the internal representations (informative indexes and/or bibliographic information, etc...). Above all, the user forbids the system to "match" some terms and so he blocks some of the rules. For such a user, the system leads the dialogue: another function of the "query (re)formulation part of this system is then to assist the user in producing complex (boolean) descriptions of the required documents. The system is useful for consultation by expert users, but it can also train the unexperienced users...

(3) A so called "specialist mode" provides tracks allowing designers (linguists, analysts, etc...) to oversee the operating processes and to break in by means of a specialized language in order to "modify the rules".

The important features, for designers and administrators, of the presently available system can be summarized as follow :

- Constant evolution in acquisition and growth of the relevant knowledge in the various domains: document descriptions, rules for search strategies, linguistic rules,...
- Learning capabilities leading to improved performances in linguistic processes and query reformulation.

Conclusion.

These ambitious efforts really bring progress. The outlook is hopeful and must be promoted to established that the method leads to a generally usable solution. But it should be necessary to have means to load a complete bibliographic database including hundred of thousands of items...

References:

[1] R. Attar, and A.S. Fraenkel, Experiments in local metrical feedback in full-text retrieval systems, Information Processing & Management, 17, (1981), 115-126

[2] J.-C. Bassano, Un système documentaire performant et son implantation, congrès AFCET Informatique, Nancy, France, (1980),

[3] J.-C. Bassano, and K. Herzallah, Composante linguistique pour un système de documentation, congrès AFCET RF-IA, Nancy, France, (1981),

[4] J.-C. Bassano, un système convivial pour la recherche documentaire, congrès RIAO 85, Grenoble, France, (mars, 1985), 28-48

[5] J.-C. Bassano, DIALECT: un système expert pour la recherche documentaire, doctorat d'Etat, Université Paris-Sud, Orsay, France, (february, 1986), 400 pp

[6] N.J. Belkin, and K. Hapeshi, Developping and evaluating an automatic on-line information retrieval system based on user problem statements, congrès RIAO 85, Grenoble, (1985), 681-698

[7] W.B. Croft, An expert assistant for a document retrieval system, RIAO 85, Grenoble, (1985), 131-150

[8] F. Debili, Analyse syntaxico-sémantique fondée sur une acquisition automatique de relations lexicales-sémantiques, Thèse d'état, Orsay Paris11, (1985), 200 pp

[9] C. Fluhr, SPIRIT: un système syntaxique et probabiliste d'indexation et de recherche d'informations textuelles, Congrès IDT 81 (ADBS-ANRT), (1982), 113-116

[10] M. Gross, Lexicon-grammars and syntactic analysis of french, Proceedingsof Coling 84, (1984),

[11] M.P. Markus, A theory of syntactic recognition for natural language, MIT Press, (1980),

[12] R.N. Oddy, Information retrieval through man machine dialogue, Journal of Documentation, 33(1), (1977), 1-14

[13] P. Pietilainen, Local feedback and intelligent automatic query expansion, Information Processing & Management, 19(1), (1983), 51-58

[14] G. Salton, H. Whu, and E.A. Fox, Un environnement automatique pour la recherche booléenne d'information, IFIP 83 (Paris), (1983),

[15] G. Salton, C. Buckeley, and E.A. Fox, Automatic query formulations in information retrieval, technical report TR 82-524, Departement of Computer Science, Cornell University, Ithaca, New York, (1982),

[16] C.O. Vernimb, Automatic query adjustement in document retrieval, Information processing & Management, 13(6), (1977), 339-353

---- NOTICE BIBLIOGRAPHIQUE ----
- DESCRIPTEURS (TERMES LIES PAR -):
BASES DE DONNEES, LANGAGE NATUREL, DOCUMENTATION AUTOMATIQUE
- AUTEUR(S) (NOM.initiales PRENOM):
BASSANO.J-C
- TITRE ET SOUS-TITRES :
ACQUISITION DE CONNAISSANCES A PARTIR DE TEXTES: UNE EXPERIMENTATION EN DOCUMENTATION AUTOMATIQUE
- LANGUE DE L'ARTICLE :
FRANCAIS
- SOURCE :
CONGRES AFCET RECONNAISSANCE DES FORMES ET INTELLIGENCE ARTIFICIELLE
79-09-12
- COMPLEMENT :
PP00-10 BIBLIOGR

- RESUME :

ON SE PROPOSE DE DECRIRE UN SYSTEME PROTOTYPE POUR DES EXPERIMENTATIONS EN DOCUMENTATION. CE SYSTEME EST ELABORE A PARTIR DE METHODES DE L'INFORMATIQUE HEURISTIQUE ET D'OUTILS MATERIELS ET CONCEPTUELS DU DOMAINE DES BASES DE DONNEES, DE LA LINGUISTIQUE, DES PRATIQUES DISCURSIVES. IL S'AGIT DE PROPOSER DES STRUCTURES DE DONNEES ET DES ALGORITHMES PERMETTANT LA CONSTRUCTION AUTOMATIQUE DE CLASSES SEMANTIQUES ET DE RELATIONS SYNTAGMATIQUES A PARTIR D'ENSEMBLES DE TEXTES, D'EN MONTRER LES POSSIBILITES D'EXPLOITATION EN RECHERCHE DOCUMENTAIRE.

! AUTOMATICALLY built reformulation rule !

1/ With the coTEXTE (- fonction VERIFIE -) : "ON SE PROPOSE DE DECRIRE UN SYSTEME PROTOTYPE POUR DES EXPERIMENTATIONS EN DOCUMENTATION..."
2/ With the ("coded")CONTEXTE (- fonction CONTROLE -):
- DESCRIPTEURS (TERMES LIES PAR -):
BASES DE DONNEES, LANGAGE NATUREL, DOCUMENTATION AUTOMATIQUE
- AUTEUR(S) (NOM.initiales PRENOM):
3/ To be matched "SYSTEME POUR (UNE) EXPERIMENTATION"

to be added "SYSTEME PROTOTYPE"
(plausibilité of 0,80)

- example of TEXT reformulation rules in the system -

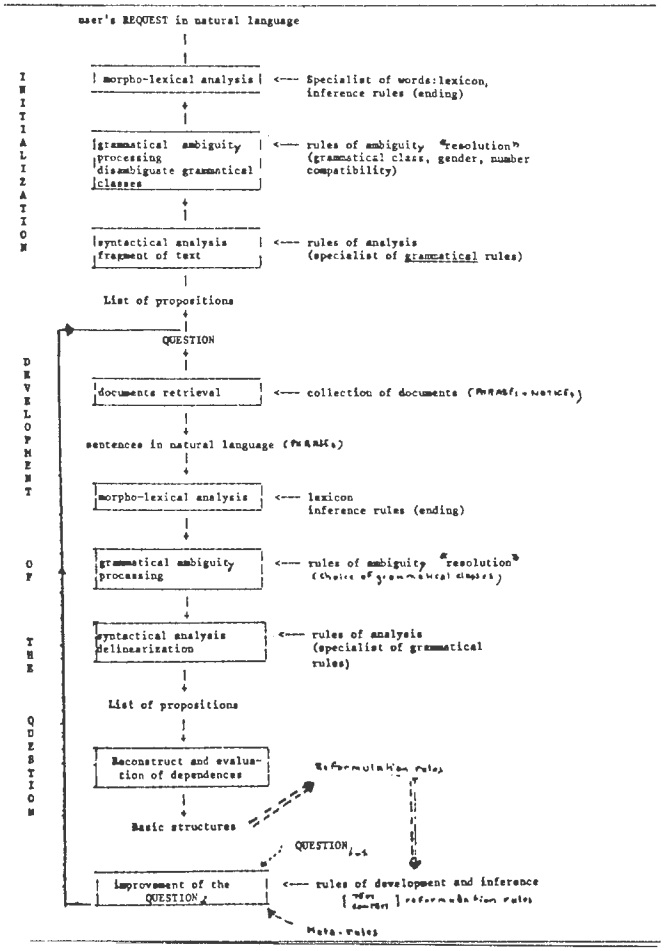
Figure 1

SI 1/ on est pour la deuxième fois consécutive en situation d'échec complet;
et si 2/ on désire poursuivre la recherche;
et si 3/ il existe des règles générales de reformulation du type INIT;

ALORS on peut choisir une stratégie (du type S4 ou "ECHEC2") qui permet, en s'appuyant sur les règles du type INIT et en tolérant lors de l'évaluation des fonctions VERIFIE des approximations plus importantes (- du type TERME -), de rechercher des "solutions approchées."

- Example of meta-rule in the system -

Figure 2



- overview of the linguistic apparatus -

Figure 4

Subdivision of Knowledge for Igneous Rock Identification*

Brian W. Otis
M.I.T. Lincoln Laboratory

Eugene C. Freuder
University of New Hampshire

Abstract

An expert system that identifies intrusive igneous rocks and the minerals that make them up is described here. One major goal of the system was to incorporate a knowledge representation in which knowledge is subdivided in a way that will increase system efficiency and inference technique flexibility, while allowing the representation to reflect as much of the original domain knowledge organization as possible. This was approached in 3 ways:

(1) Two cooperating expert systems were developed which divided the task into smaller, more manageable subsets. This is shown to increase both system efficiency and inference flexibility.

(2) Knowledge is subdivided using a two dimensional network of nodes with each node containing information about a subclass of the entire domain.

(3) Processing of knowledge corresponding to relevant subclasses is handled by three separate inference engines, each working on its own aspect of the igneous rock identification process.

1. Introduction

At least three major reasons exist why the subdivision of knowledge is an important issue in the design of any expert system. The most obvious is that of efficiency. Because the power of most expert systems lies in the amount of knowledge they incorporate, systems tend to grow quite large. Good knowledge subdivision can greatly reduce the time it takes to process this information.

Effective knowledge subdivision should also allow modularity that is inherent in the domain to be reflected in the knowledge representation. A major emphasis has been put on building expert systems that reason so that humans can easily observe and confirm the strategy used. This process is enhanced if the knowledge is subdivided in a manner which maintains the natural organization of the domain.

The third benefit of good knowledge subdivision has to do with inferencing that must take place in any expert system. Solutions to specific problems, even within the same domain, are often best attained with very different inference mechanisms. Subdividing knowledge efficiently allows the writer of the system to make use of the inference techniques that best suit the nature of particular subproblems.

Igneous rock identification is a domain in which all these aspects are relevant. This paper describes how these and other aspects of knowledge representation are handled in a system that identifies intrusive igneous rocks and the minerals that make them up. In addition, an attempt is made to identify generic aspects within the knowledge representation which may indicate techniques that could be useful in other domains.

2. Knowledge Subdivision Literature

The technique described here utilizes major domain features to subdivide knowlegde of an entire domain into nodes, or subclasses, and further investigates interesting nodes by processing small rule sets associated with those nodes. Other research has addressed similar problems.

One system, PUFF [1][14], is itself a straightforward production system for pulmonary function diagnosis but its domain has been used as a test bed for two systems with interesting, subdivided, knowledge representations. CENTAUR [1], an implementation of PUFF, also uses rules to represent important medical information but represents control and context with frames. This allows the rules to be subdivided into groups, each group containing information about a specific problem. A group is associated with a frame and since the control of the system is a function of which frame is being processed, control need not be handled by the rules.

Smith and Clayton [14] take this a step further in their system, WHEEZE, by making more extensive use of frames. They do this by using frames not only to represent the control and context but also to represent the domain knowledge. Both CENTAUR and WHEEZE encompass the same medical information as PUFF. Their different representations result in a useful contrast of subdivision techniques using frames and rules.

In GUIDON [6], an attempt to use a Mycin-like rule base in a tutorial system, the tutorial aspect forces an interesting knowledge base subdivision of a different type. In this system, domain and teaching knowledge are separated and the domain knowledge is further subdivided into a three tier representation, meta-level abstraction, performance, and support.

Two systems that have used semantic networks to enhance knowledge subdivision are Internist [13] and Prospector [7]. Internist is a specialist in internal medicine in which a semantic network knowledge representation is used to represent medical knowledge in the form of a hierarchical disease tree. Terminal nodes in the tree represent actual diseases. Nonterminal nodes and their subtrees represent particular disease areas. Nodes are associated with disease manifestations. Processing compares the initial manifestations entered by the user and other information to the network, deriving the most probable disease or diseases.

Prospector [7], a system which predicts the likelihood that a region will contain certain ore deposits, and the subsequent knowledge acquisition system that evolved from it,

KAS, both use semantic networks to break down their knowledge bases. The semantic networks in Prospector represent the models of potential ore depostis against which the system compares data input by the user. KAS extracts the domain independent aspects of this process.

The concept of using features to narrow the domain has been explored for pattern recognition applications [3]. The idea is to extract the most important features to build a feature space. This feature space contains a compact description of the relevant properties. Properly chosen features prove very helpful in pattern recognition and poorly chosen features may be useless. Ballard concludes that good feature selection and determination are essential for good pattern recognition.

Finally, William Clancey [6] takes a very high level look at the general characteristics of the knowledge needed for classification problem solving. Instead of giving specific representations, he attempts to identify the program flow, data abstractions, and heuristics that are usually present in any classification system. This discussion lends some insight to what type of knowledge subdivision may generally be desirable for a classification system.

Characteristics of intrusive igneous rock identification results in a unique knowledge subdivision and processing strategy combination. This paper describes the resulting subdivided knowledge representation. The mineral and rock identification system uses features along a two dimensional grid to break the entire domain into subclasses. This strategy allows the system to process rules associated with only the most promising nodes, resulting in efficient, effective reasoning.

3. The Mineral and Rock Identification System

The complex nature of many Geological domains has led to their being the focus of several interesting expert systems [4, 5, 6, 8, 13]. The system described here is a two phase expert system that will aid the user in the identification of minerals and rocks. An interaction, led by the system, questions the user about the physical properties of a specimen and subsequent processing leads to the system providing the most probable rock or mineral identification. This system is implemented on a VAX 11-780 in Franz Lisp. It makes use of a Franz Lisp extension called GRASPERUNH [2], an implementation of a graph representation language originally developed at the University of Massachusetts [10].

4. Overview of Representation Techniques

Along with many fairly common expert system building techniques, this system incorporates techniques designed to facilitate a knowledge subdivision that handles the three issues mentioned above. The combination of these techniques are aimed at maximizing system efficiency with respect to actual processing time, providing a logical representation that reflects the natural domain organization, and making it easy to incorporate different inferencing strategies.

The first of these techniques is a two phase identification process which is useful because of the atomic/molecular relationship which exists between minerals and rocks. Since rocks are composed of minerals the ability to identify minerals is a prerequisite for rock identificaion. Reference to a "mineral expert" is often needed during the rock identification process. To satisfy this, phase one of the system is itself an expert system for mineral identification and phase two, using phase one if needed, is an expert system for rock identification. The mineral identification system may be used as a front end to the rock identification phase and it may be consulted during rock identification.

This division is of interest not only because of the communication between phases but also because it represents a natural break of the entire problem into problems that require different inference mechanisms. As will be described later, the rock identification problem is quite different than the mineral identification problem. A clean break here allows the system to utilize the problem solving strategy best suited to the nature of each problem.

Other interesting knowledge subdivision concepts are incorporated within the rock identification phase. The basis for the knowledge subdivision here is the three-fold. The first demonstrates the idea that we would like the modularity in the knowledge to reflect the modularity in the domain itself. Because rocks can be grouped into classes, it is possible to associate subsets of the knowledge with certain subclasses of rocks.

The second criteria is that these subsets of knowledge, which represent subclasses of rocks, can then be further broken down. This breakdown corresponds to different features of the rock identificaton problem, including refining the rock name, determining the rock's texture, and identifying accessory minerals. In general any inference mechanism could be applied to each subproblem. In this system, the problems of refining the rock name and determining the rock's texture are both accomplished through forward chaining. Accessory mineral identification utilizes the users input combined with possible assistance from the mineral identification system.

The final characteristic of the domain that has a major impact on the knowledge subdivision is that complications are encountered when an identification system has, as its domain, items that continuously grade into one another. Because rock compositions may often be continuously gradational with respect to mineral percentages and textures, the process of accurately determining a rock's identity may be complex. In fact boundaries determining where one rock type stops and another begins are often artificial. The result is a taxonomy-like breakdown of discrete entities superimposed on items that in the real world actually grade continuously into one another. Although identifying a subclass on which to focus allows the system to isolate subsets of relevant knowledge, information regarding similar subclassses must be close at hand. This is because the continuous gradation of items in the domain increases the possibility that the incorrect subclass has been chosen when dealing with items near the artificial boundaries.

The resulting architecture is an attempt at segmenting the domain by setting up a two dimensional grid network with a gradational property or properties along each axis. Nodes within the network represent a subset of similar rocks and are activated according to where a specimen falls compositionally along each of the two axes. This allows knowledge sources to continue further rock identification processing using only a limited subset of the entire knowledge base. If more than one node is activated, rules within nodes may disprove the node itself while either lending evidence to previously activated nodes or activating new nodes.

All the aspects described above have positive impact on our initial, most obvious reason given for good knowledge subdivision, efficiency. Successful identification of the

relevant information allows the system to spend most of its time on important processing, thus decreasing the amount of time it takes to make a final assessment.

5. System Architecture Overview

Although a detailed understanding of the entire system is not necessary to appreciate the knowledge representation, a top level understanding of the control flow used in the recognition process is useful. Figure 1 shows the architecture of the entire system. The upper left represents phase one, the mineral identification system, and the rest is phase two, the rock identification system. Each time the system is started it is in the state represented by box #1. Note that the mineral identification phase is not only a front end for the rock identification phase but it may also be consulted during rock processing.

As Figure 1 shows, the mineral identification phase combines forward and backward chaining to make its identification. The rest of the figure shows the rock identification phase which goes through a series of node activation levels to determine which nodes are most likely to contain the proper rock group. Nodes are then processed using three knowledge sources, each an expert on one aspect of rock identification.

6. Representation Details

This section contains a detailed description of the knowledge representation implemented in this system. Keeping the pieces in order with respect to Figure 1 may be helpful. Emphasis in this system was given to the major representation issues including: (1) Interaction between expert systems

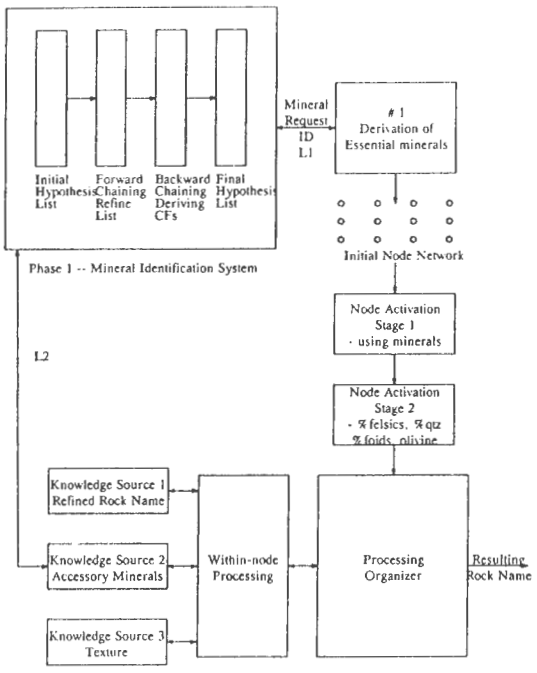


Fig. 1. Entire system architecture.

demonstrated by the rock identification phase using the mineral identification phase, (2) the two dimensional grid structure of nodes for subdividing the knowledge needed to represent a continuous, gradational domain with a superimposed artificial taxonomy (3) a strategy to process this grid of nodes leading to high efficiency and accurate results, and (4) knowledge sources used to solve different aspects of one problem, as demonstrated when individual nodes are processed to determine refined rock name, accessory mineral assemblages, and texture description. This paper will discuss only the features which had the most substantial impact on system design, features (2), (3) and some aspects of feature (4) described above. More extensive system details can be found in [12].

6.1 Segmented Knowledge Representation in a Gradational Domain

Although essential mineral composition does not result in complete rock identification it is a good place to start. Other features that come into play such as refining the mineral assemblage, texture, and accessory minerals, will be discussed later. Figure 2 from [15] is an example of the diagrams used by geologists to represent the compositional variations in the intrusive igneous rocks. It shows a pair of triangular phase diagrams. Their details are not important for this discussion except for the fact that each triangle represents a continuous solid solution series between the components found at the three triangle vertices. To locate a rock group name, the compositional percentage of each of the three components must be plotted along the three axes of the triangle. The result is a point within the triangle which represents the rock's essential mineral composition.

The important thing to note from this diagram is that the triangles each represent a continuous gradation between three major components. For classification purposes, geologists have superimposed an artificial taxonomy on top of these triangular phase diagrams. These are indicated by the lines within the diagram and the rock group names such as granite, monzonite, syenite, etc. Although geologists' reasons for implementing such a scheme are obvious, it can be misleading because breaks in the taxonomy are distinct even though the gradation is continuous. During the identification, processing must control for the fact that these distinct breaks are arbitrary in a continuous real world where rock groups abut one another with respect to composition.

The fact that rocks compositionally close to boundaries will be very difficult to place in one group or another must be taken into account. If the artificial boundaries are to be of any help in the subdivision of knowledge it must be done very carefully making sure all possibly relevant information is included in the processing. The problem of representing the information contained in several of these diagrams in a way that would lend itself to efficient and effective processing turned out to be a major design issue for this system.

6.1.1 Segmented Organization The major subdivision for rock identification breaks all the rocks within the domain into subclasses. The features used to determine this subdivision should be the features of the domain that can be most easily distinguished by the system user. The triangular phase diagrams often have axes that can be more easily determined than others. Because of this, it is desirable that the major subdivision is based on the more easily determined features. This would give the greatest possibility that the correct subclass would be chosen for processing.

The technique chosen was to extract a few major features from the domain that allowed a good deal of the information to be represented in a two dimensional form. Figure 3 is the result of this process.

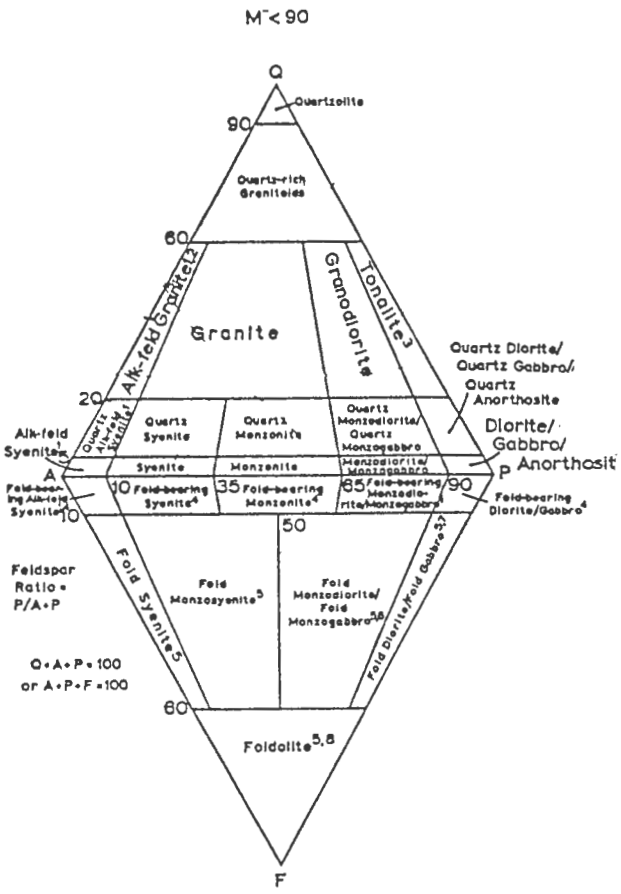


Fig. 2. Classification of Igneous rocks.

Some of the ideas for which features may be most important and most easily determined come from [11]. This two dimensional graph is a form that can lead to an efficient processing technique and may be applicable to the breakdown of other domains. This organization turned out to resemble the basic pattern recognition feature space technique [3].

The y-axis represents the percentage of all the minerals in the rock that are felsic minerals. This is a commonly used identification feature of igneous rocks and its determination is not dependent on any mineral identifications. The x-axis is slightly more complex and at first glance appears multidimensional in itself. However, close examination shows that its major features are mutually exclusive. These features are the percent of the felsic minerals that are quartz and the percent of the felsic minerals that are feldspathoids. As shown in Figure 2 these minerals normally do not occur together thus the placement of these two features on one axis is valid. When the two minerals do occur together they are both present in small percentages and this case is accounted for in column 5 of Figure 3. The third criteria found on the x-axis is the presence or absence of olivine in the specimen.

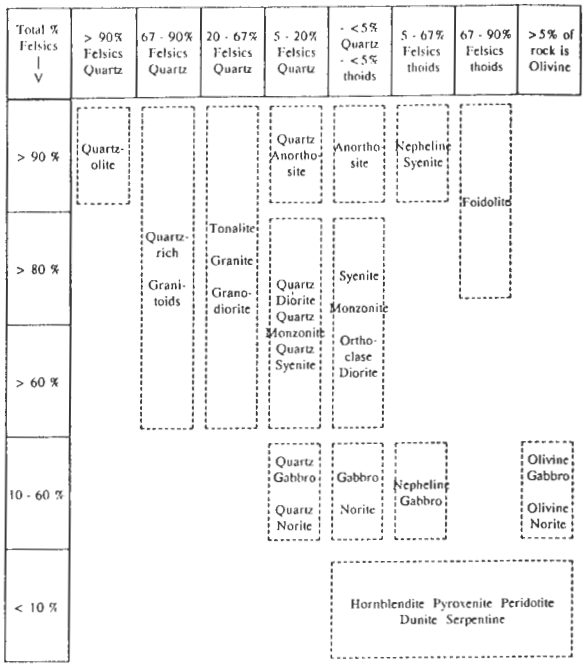


Fig. 3. Initial network of nodes for subdivision of rock domain.

Carefully plotting each of the major intrusive igneous rock groups on these two axes results in the group names within Figure 3. It should be noted that, for the most part, the details of where the rock groups fall on the y-axis cannot be determined from Figure 2 and other sources must be used [11]. Careful analysis of the rock groups allows for combining groups into "nodes" of groups, or subclasses, which cannot be completely separated using the information contained on the x and y axes.

Because a major criteria for axes labels was the ease with which the features could be distinguished, a new, intermediate taxonomy is created by the nodes shown on the graph. This new taxonomy is slightly coarser grained than the original defined by geologists. "Fine grained" classification is accomplished by processing knowledge within the nodes of this new taxonomy.

Determination of node breakdown is simple in some cases and more complex in others. For example, from Figure 2 one can see that quartzolite is the only rock type whose felsics are greater than 90 percent quartz. This rock type is also normally greater than 90 percent felsic total, logically placing only one node in column 1.

Columns containing greater than one rock type require further investigation. Column 3, for example, shows one node with three different rock types. Figure 2 shows that the major feature that separates tonalite, granite, and granodiorite is the orthoclase/plagioclase feldspar ratio. Since this feature is not directly represented on either axis of the graph, differentiation of this property must take place during within-node processing. Columns four through eight, on the other hand, all have at least two groups of types. Groups are separated only when they can be distinguished according to criteria represented on an axis. Similar analysis of the entire diagram results in the node breakdown shown in Figure 3 (dashed boxes). Once nodes are created they are connected by arcs to all neighboring nodes and any nodes that contain

similar rocks allowing them to reference one another during within-node processing. This process will be described in more detail below.

It is interesting to note that the only node that crosses vertical boundaries is the one containing hornblendite, pyroxenite, peridotite, dunite and serpentine. The total felsics in these rocks is less than ten percent. Some of these rocks contain olivine, and as a whole form a continuous solution series shown in its own triangular phase diagram. In this case a distinct break cannot be formulated by the criteria along the x-axis and the node must be created as shown. In all other cases the man made divisions are well defined according to the x-axis divisions. This, of course, does not guarantee that the user can give exact enough information to insure perfect placement. In the case that he cannot, multiple nodes may be activated and the distinction, if possible, is made according to other criteria during within-node processing.

This two dimensional representation was chosen for two major reasons. First, it was decided that in order to obtain acceptable performance, only major, easily distinguished features should be used for the initial, coarse grained classification. Secondly, the less complex a knowledge representation is, the more easily can be understood by the domain expert. A more complex representation, one using greater than two dimensions for example, was undesirable because it would have brought minor features into the initial breakdown and it would also have made it harder for the domain expert to understand the representation.

6.1.2 Network Processing Strategy Processing of the network developed above, and shown in Figure 3, proceeds in two stages during a rock identification, node activation and within-node processing. Node activation is itself a two stage process. First, the user's initial, essential mineral list is compared to mineral lists contained in each node. If the initial list derived from this process indicates the node could possibly be the correct node it is added to a list which goes to the second activation stage.

The second activation stage asks questions to provide necessary system information to determine where, on each axis, the specimen falls. If the user indicates, or the system determines, that the input is not accurate enough to narrow processing to one section of an axis, more than one section will be included in the activation process. Nodes that fall on the intersection of any valid x and y combination that were also activated during the mineral comparison are put on an active node list.

Although it is possible for nodes that are not activated by this initial activation to become involved during with-in node processing, it is hoped that using the combination of essential minerals and the most easily identified major features, will result in an accurate initial list of activated nodes.

6.1.3. Knowledge Source Organization for Within-Node Processing Active nodes are then passed to within-node processing. Each active node has a priority number associated with it that indicates how well the node's minerals matched the minerals used during stage 1 activation. The active node list is ordered so the nodes that matched best will be processed first. This order may or may not have an important impact on processing, depending on the mode of the system. The user has the option of setting a switch that will cause the system to exit when the first successful node is encountered. On the other hand, the user may force the system to process all active nodes, in which case the order is not as important.

During within-node processing, narrowing the subclass to a final rock determination is done by three knowledge sources. The inference engine for each knowledge source is a separate entity. Each node contains rules, or other information, tailored to that node for each knowledge source. It should be noted that the term knowledge source is used here to refer to the inference mechanism and its associated information in the nodes. Although this may be slightly contrary to the usual concept of a knowledge source, the term is used because the knowledge sources used here fulfill the same purpose that is usually associated with knowledge sources, to act as an expert on a small subproblem within the entire domain.

Upon activation of a knowledge source, theinformation from the current node that corresponds to that knowledge source, is passed in for processing. Individual knowledge sources deal with (1) refining the rock name, (2) accessory minerals present, and (3) rock texture.

This breakdown in both the processing and the knowledge is effective for three reasons. First, the obvious knowledge subdivision which allows only the subset of information that is relevant to the given problem is considered. Second, each of these knowledge sources can utilize an inference mechanismbest suited for the particular subproblem. Third, the rules that would be necessary in a strict rule based system to handle control are not necessary because they are inherent in the structure. This means that subsets of rules for a given knowledge source must only have rules for domain expertise on that specific subproblem. This is an advantage over strict rule-based system where control rules and rules for other subproblems may confuse the rule set. A more sophisticated version of the knowledge source - blackboard architecture is found in the Hearsay systems [8].

7. Domain Independent Representation Implications

One goal of this work is to see if any of the knowledge representation concepts utilized for this domain could be abstracted into a domain independent form. There seems to be two possible results from this analysis, one fairly obvious, very high level scheme, and one more detailed technique.

The obvious case is that when identifying objects with an atomic/molecular relationship, a pair of experts may be desirable. It allows for the abstraction of many details concerning the atomic identification during the molecular identification. This thought can be taken one step further by saying that as expert systems of all types grow in complexity it may be logical to build sets of conferring expert systems to modularize the processing and the large amounts of knowledge.

The second possible domain independent representation technique is that demonstrated by the two dimensional network of nodes. It appears that this network architecture, and the mechanism described for processing it, may be a useful representation technique for other domains. The domain should have features across which it can be subdivided. In general, if such a domain has been identified, the procedure is to:

(1) Extract two (or more if mutually exclusive features exist) general features and represent them on the axes of a two dimensional grid.

(2) Locate each major item type of the domain of the grid.

(3) Bundle item types which cannot be differentiated strictly by features on the x and y axes into "nodes".

(4) Connect neighboring nodes. If further within-node processing excludes a node it can use this connection to access the node most consistent with the new information.

(5) Differentiate items within a node by examining features that are not represented on the x or y axis and by taking a closer look at features tht are represented on the grid.

8. Summary/Conclusions

This system demonstrates techniques to subdivide knowledge that works well in the domain of igneous rock identification. Features of this domain that make these techniques applicable probably exist in other domains, thus, the use of these techniques may prove beneficial for systems in these other domains.

First, because of the atomic/molecular relationship that exists between minerals and rocks, a two phase system is discussed. Phase one is a stand alone mineral identification expert system. Phase two is a rock identification expert system which has easy access to phase one. This combination allows the user to access the mineral expertise from phase one at any time during rock identification. This split represents a natural break in domain knowledge as well as a break between two separate problems that can be solved by very different processing strategies.

The second feature is a knowledge representation that allows encoding knowledge in a continuously gradational domain with an artificial, superimposed taxonomy. This representation allows major features to be represented on the axes of a two dimensional grid and nodes to be developed within the grid. Processing includes activating nodes and within-node processing that narrows the possibilities down to a final rock description. Along with allowing logical rule group divisions this modularization of knowledge increases processing speed by narrowing the entire knowledge base into smaller subsets of rules and information that must be considered for any one identification.

The subdivision techniques described here work well for the given domain. Although systems tend to become molded for a particular domain, proof of domain independence would be the true test of the architecture's value. This, of course, can be accomplished only through its application to systems in other domains.

9. References

[1] Aikins, J.S. **A Representation Scheme Using Both Frames and Rules in Rule-Based Expert Systems,** Addison-Wesley, 1984.

[2] Arcidiacono, T., **A Graph Representation and Processing Extension to Franz Lisp,** Master's Thesis, University of New Hampshire, 1985.

[3] Ballard, Dana H. and Christopher M. Brown, **Computer Vision,** Prentice Hall.

[4] Barr.A., and E.A. Feigenbaum, **The Handbook of Artificial Intelligence,** Volumes 1 and 2, William Kaufmann, Inc., 1981.

[5] Bonnet, A. and C. Dahan, **Oil-well Data Interpretation Using Expert System and Pattern Recognition Techniques,** Proceedings of IJCAI-83, Volume 1, 1983.

[6] Clancey, W.J. **Classification Problem Sovlign,** Proceedings of AAAI-84 Volume 1, 1984.

[7] Davis, R., H. Aussin, I. Carlbom, B. Frawley, P. Prochnik, R. Sneiderman, J.A. Gilreath, **The Dipmeter Advisor: Interrelation of Geologic Signals,** Proceedings of IJCAI-81, Volume 2, 1981.

[8] Duda, R., J. Gaschnig, and P. Hart, **Model Design in the Prospector Consultant System for Mineral Exploration,** Readings in Artificial Intelligence, Webber, Nilsson, Tioga Publishing Company, 1981.

[9] Erman, L.D., Victor R. Lesser, Reddy, D. Raj. **The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty,** Computing Surveys, Vol. 12, no. 2, June 1980, pp. 153-213.

[10] Gachnig, J.P., **Application of the Prospector System to Gological Exploration Problems,** in Machine Intelligence 10, Hayes, Michie and Pao, 1982.

[11] Lowrance, J.D., **Grasper 1.0 Reference Manual,** COINS Technical Report 78-29, University of Massachusetts/Amherst, 1978.

[12] Moorhouse, W.W., **The Study of Rocks in Thin Section,** Harper and Row, 1959.

[13] Otis, B.W., **An Expert System to Identify Rocks and Minerals: A Case Study in the Subdivision of Knowledge,** Master's Thesis, University of New Hampshire, 1985.

[14] Pople, H.E. Jr., **The Formation of Composite Hypotheses in Diagnostic Problem Solving, An Exercise in Synthetic Reasoning,** IJCAI-77, Volume 2, 1977.

[15] Smith, R.G. and J.D. Baker, **The Dipmeter Advisor System,** A Case Study in Commercial Expert System Development, Proceedings of IJCAI-83, Volume 1, 1983.

[16] Streckeisen, A.L., **Ecology,** vol. 7, pg. 354-355, 1979.

* This work is sponsored by The University of New Hampshire. The views expressed are those of the author and do not reflect the official policy or position of the U.S. Government.

A Hybrid, Decidable, Logic-Based Knowledge Representation System

Peter F. Patel-Schneider

Schlumberger Palo Alto Research
3340 Hillview Avenue
Palo Alto, CA 94304

ABSTRACT

The major problem with using standard first-order logic as a basis for knowledge representation systems is its undecidability. A variant of first-order tautological entailment, a simple version of relevance logic, has been developed that has decidable inference and thus overcomes this problem. However, this logic is too weak for knowledge representation and must be strengthened. One way to strengthen the logic is create a hybrid logic by adding a terminological reasoner. This must be done with care to retain the decidability of the logic as well as its reasonable semantics. The result is a stronger decidable logic that can be used as the basis of a knowledge representation system.

A knowledge representation system is supposed to store facts about areas of the real world and to retrieve them and consequences of them in a reasonable amount of time. This description of knowledge representation systems seems innocuous at first, but has several important consequences. First, a knowledge representation system cannot really store facts at all, but only expressions in some formal language. To say that it stores facts, there must be some external semantics for its expressions, one which makes their meaning plain. This semantics then serves as the specification of the knowledge representation system.[1] Second, to represent a large number of areas of the real world, a knowledge representation system must have adequate expressive power. Third, to discover facts inherent in the stored facts, this semantics must have a powerful deductive capability. Fourth, because timely responses are required, this deductive capability must be computationally tractable.

The usual way for providing an external semantics to a knowledge representation system is to use some formal logic with a model theoretic semantics. Operations of the knowledge representation system then correspond directly to some aspect of the semantics, such as question answering in the knowledge representation system corresponding to determining logical consequence in the semantics.[2] Model theoretic semantics forms the only reasonable way of providing an external semantics to a knowledge representation system.

To achieve expressive power adequate to represent facts in many situations of interest to knowledge representation, conjunction, disjunction, negation, and quantification all seem to be necessary. Moore [10, p. 428] convincingly argues that "a general-purpose representation formalism [needs (at least)] all the features of first-order classical logic with equality". This, along with the well-developed and intuitively pleasing semantics of standard first-order logic, and the resulting deduction system, makes standard first-order logic the logic most used for knowledge representation.

Unfortunately, standard first-order logic has a severe problem—determining whether one formula follows from another in it is, in general, undecidable. Therefore, a knowledge representation system that satisfies the first three requirements above cannot guarantee timely responses, or even guarantee that it will produce responses at all.

There have been many efforts to create knowledge representation systems that use standard first-order logic but also are decidable and thus better satisfy the fourth requirement. Some of these systems simply terminate processing in some *ad hoc* manner, which means that no complete characterization of this processing can be given in the semantics, thus violating the first requirement. Other knowledge representation systems use the syntax of standard first-order logic, but limit deductions by using a complicated semantics that includes syntactic structures, so that the semantics, which should be concerned with the truth and falsity of formulae, instead simply models the deductive process. The semantics then does not provide a meaning for formulae, and again these systems violate the first requirement.

A further possibility is to use a different logic, with a different model theoretic semantics and, possibly, a different syntax. The goal would be to produce a logic with decidable inference but also with the expressive power of standard first-order logic and a motivated model theoretic semantics.[3] This logic would be weaker in deductive power than standard first-order logic, but this is the price that will have to be paid to achieve decidability. Inference in the logic should also be fast enough in normal circumstances so that a knowledge representation system built on it could be treated as a black box, with no need for extra-logical control of the system, as is needed in all knowledge representation systems based on standard first-order logic.

Some interesting work using different logics has been done for propositional logics, such as the work on propositional tautological entailment (a particularly simple type of relevance logic) by Levesque [8]. Propositional tautological entailment has an intuitive four-valued semantics, and determining whether one formula in conjunctive normal form follows from another can be done in time proportional to the product of their sizes. This contrasts with the situation in standard propositional logic, where this problem is co-NP complete. What is lost by going to propositional tautological entailment is a great deal of the deductive power of standard propositional logic, particularly *modus ponens*. However, this shows that there are logics with reasonable semantics for which reasoning is considerably more tractable than in the standard logics. The problem with propositional logics, is that, lacking quantification, they are just not expressive enough

[1] This emphasis on formal accounts of meaning for knowledge representation systems in particular and AI programs in general has been argued by various people at various times over the life of AI. Some important papers in this vein are [7], [10], and [6].

[2] The use of logic as a basis for knowledge representation systems is largely due to these well-defined model theoretic semantics.

[3] Most of the non-standard logics used in AI, however, have been proposed to solve different problems. Non-monotonic logics, default logics, and several other logics used in AI were designed to augment the power of standard first-order logic in some way and also suffer from being undecidable.

for knowledge representation.

Other versions of relevance logic have been proposed as possible logics for knowledge representation. Following arguments by Anderson and Belnap [1, pp. 1-30] that relevant implication models implication better than material implication does, Shapiro and Wand suggested using a version of relevant implication as the basis of a knowledge representation system [13]. A particular version of first-order relevant implication was axiomatized and used as the basis of a knowledge representation system [9]. However, even though the version of relevance logic used in this system is weaker than standard first-order logic, it is still undecidable. Belnap [2] also suggested that relevance logic be used for knowledge representation. The version of relevance logic he used is related to tautological entailment, but is more complex. Again, this logic, although weaker than standard first-order logic, is undecidable.

The arguments in the above papers in favor of relevance logic as a logic for knowledge representation indicate that a decidable first-order relevance logic would be very useful as the basis of a knowledge representation system. The result mentioned in [8], concerning the tractability of reasoning in propositional tautological entailment, suggests that first-order tautological entailment might be decidable: however, first-order tautological entailment can be used to simulate standard first-order logic and is thus not decidable. In order to obtain a decidable first-order relevance logic, it is necessary to further weaken first-order tautological entailment.

A decidable variant of first-order tautological entailment, where quantification is no longer equivalent to infinite conjunction or disjunction, has been developed [11]. This logic has a reasonable four-valued semantics and is suitable as the basis of a knowledge representation system. The problem with this logic, and also with first-order tautological entailment, is that it is extremely weak. In particular, given that Tweety is a bird and that all birds are animals, $bird(Tweety)$ and $\forall x\ \neg bird(x) \vee animal(x)$[4], it is unable to conclude that Tweety is an animal. This weakness is necessary to achieve a simple decidable logic, but the logic is too weak to use as the direct basis of a knowledge representation system.

I Hybrid Logics

What is needed is some way of modifying the logic so that more inferences are supported, while still retaining decidability. One general method of modifying a logic to increase its power is by using hybrid logics. A *hybrid logic* is created by taking two logics, the *base logic* and the *auxiliary logic*, that share the same semantic structures, and combining them in a certain way.

The base logic is a logic composed of a syntactic language of sentences, a class of semantic structures, a set of truth values with a partial order on this set, and an interpretation function (usually represented by $i(\alpha, S)$) for deriving truth values from semantic structures and sentences.[5] (This unusual way of specifying truth values allows for the four-valued logic that will be used later.) One sentence, β, follows from another, α, in the base logic if, for all semantic structures S, $i(\alpha, S) \leq i(\beta, S)$.

The auxiliary logic is just another logic, but it is easier to understand the combination of the two logics if the semantics of the auxiliary logic is defined in a slightly different manner. The auxiliary logic consists of a syntactic language of sentences, the same semantic structures as those of the base logic, and an interpretation function that specifies which semantic structures satisfy which sentences of the logic. A sentence in the auxiliary language, A, follows from a set of sentences in the auxiliary language, $\aleph$, if, for all semantic structures, S, if S satisfies all elements of $\aleph$ then S satisfies A.[6]

Logical consequence in the hybrid logic combines both of the logics. One sentence of the base logic, β, follows from another, α, in the presence of a set of sentences from the auxiliary language, $\aleph$, if, for all semantic structures S, such that S satisfies each element of $\aleph$, $i(\alpha, S) \leq i(\beta, S)$.

One example of a hybrid logic is the logic used by KRYPTON [5]. Here, the base logic, called the assertional logic, is standard first-order logic and the auxiliary logic is a terminological logic of definitions of predicates. A semantic structure satisfies a terminological definition in KRYPTON when the objects that satisfy the defined predicate are precisely the same as those that satisfy its definition. This use of the auxiliary logic to define predicates is somewhat similar to, but more powerful and better defined than, the way concepts are defined in semantic networks.

The hybrid logic of KRYPTON has the same deductive power as standard first-order logic because representing the definitions as universally quantified bi-conditional statements and adding them to sentences of the assertional logic results in the same assertional inferences as those sanctioned by the hybrid logic. However, the terminological language is interesting in its own right and serves the important role of distinguishing predicate definitions from assertional statements. Further, the hybrid system runs faster than the equivalent non-hybrid system, an important issue for knowledge representation.

The utility of the terminological logic in KRYPTON suggests that adding a terminological logic to the decidable variant of first-order tautological entailment might be a useful way of augmenting its utility for knowledge representation. The difference between this hybrid logic and the hybrid logic of KRYPTON is that expressing the definitions as universally quantified bi-conditional statements would not result in the same deductive power.

II The Assertional Component

Since both components of a hybrid logic share the same semantic structures, a short description of the assertional component, focusing on its semantics, is required before the terminological component can be defined.[7]

The syntax of the assertional component is the same as that of first-order tautological entailment, and thus almost the same as standard first-order logic. The semantics of this variant starts with first-order situations, the analogue of first-order models. A *situation* consists of a non-empty set D, the *domain* of the situation, and two mappings, h and j. The mapping h is a mapping of function letters into functions over the domain. The mapping j maps each predicate letter, A^n, into a function from D^n to subsets of $\{t, f\}$ and determines the truth value of atomic formulae as shown below. The complexity in j is needed because an atomic formula can be assigned true, false, neither, or both. This change from two to four truth values means that there are many

[4] It is possible to argue that this is a bad translation of "all birds are animals" but it is the only way provided in first-order logic.

[5] For standard first-order logic the language is the usual language of sentences, the semantic structures are Tarskian interpretations including mappings from predicate symbols to relations and from function symbols to functions, the truth values are "true" and "false" with "true" being greater than "false", and the interpretation function is the usual one.

[6] This change from using single sentences in the base logic to using sets of sentences in the auxiliary logic is not crucial but does help when talking about hybrid logics using terminological logics as their auxiliary logic.

[7] For a more detailed description, including proofs of the theorems in this section, see [11].

more situations than possible worlds, and possible worlds correspond to those situations with no "missing information" and no contradictions.

Variable maps map variables into elements of some domain in the usual way. If ν is a variable map into D, x is a variable, and d is an element of D, then ν_d^x is a variable map into D with $\nu_d^x(y) = d$ if $y = x$, and $\nu_d^x(y) = \nu(y)$ otherwise. Variable maps can then be extended, again in the usual way, into mappings for arbitrary terms. Given a situation, s, and a variable map, ν, into the domain of s, the mapping ν_s^* is defined as $\nu_s^*(x) = \nu(x)$ if x is a variable, and $\nu_s^*(f^n(t_1, \ldots, t_n)) = (h(f^n))(\nu_s^*(t_1), \ldots, \nu_s^*(t_n))$ otherwise.

Next comes the definition of *compatible sets of situations*. A compatible set of situations is a set of situations with the same domain and the same mapping of function letters to functions. In other words, the situations in a compatible set of situations differ only on their interpretation of atomic formulae. In this logic the meaning of formulae will be determined with respect to these compatible sets of situations, and not simply with respect to single situations, as in regular first-order tautological entailment.

Given S, a compatible set of situations each with domain D, and ν, a variable map into D, the interpretation function for the logic is defined as follows:

1. $\mathrm{t} \in i(A^n(t_1, \ldots, t_n), S, \nu)$ iff
   for all $s \in S$ $\mathrm{t} \in j_s[A^n](\nu^*(t_1), \ldots, \nu^*(t_n))$
   $\mathrm{f} \in i(A^n(t_1, \ldots, t_n), S, \nu)$ iff
   for all $s \in S$ $\mathrm{f} \in j_s[A^n](\nu^*(t_1), \ldots, \nu^*(t_n))$
2. $\mathrm{f} \in i(\neg\alpha, S, \nu)$ iff $\mathrm{t} \in i(\alpha, S, \nu)$
   $\mathrm{t} \in i(\neg\alpha, S, \nu)$ iff $\mathrm{f} \in i(\alpha, S, \nu)$
3. $\mathrm{t} \in i(\alpha \wedge \beta, S, \nu)$ iff $\mathrm{t} \in i(\alpha, S, \nu)$ and $\mathrm{t} \in i(\beta, S, \nu)$
   $\mathrm{f} \in i(\alpha \wedge \beta, S, \nu)$ iff
   $\exists S_1 \cup S_2 = S$ $\mathrm{f} \in i(\alpha, S_1, \nu)$ and $\mathrm{f} \in i(\beta, S_2, \nu)$
4. $\mathrm{t} \in i(\exists x\alpha, S, \nu)$ iff for some $d \in D$ $\mathrm{t} \in i(\alpha, S, \nu_d^x)$
   $\mathrm{f} \in i(\exists x\alpha, S, \nu)$ iff for all $d \in D$ $\mathrm{f} \in i(\alpha, S, \nu_d^x)$

The interpretations of disjunction and universal quantification are determined by using the usual equivalences.

Under this semantics $\exists x\alpha$ is true in S under variable map ν if there is some domain element, common across all the situations in S, which, when taken as the mapping of x, makes α true in each situation. The same formula is false if all domain elements, when taken as the mapping of x, make α false in each situation. For quantifier-free formulae, this semantics is equivalent to the semantics for regular first-order tautological entailment.

One side-effect of this change in the meaning of quantification is that quantifiers cannot be moved around and combined in all the ways they can in standard first-order logic while maintaining equivalence. Further, there are formulae which cannot be converted into an equivalent prenex form by simply moving quantifiers.

Now that the support relations have been defined, the notion of entailment can be defined in this logic. As it turns out, there are three different versions of entailment possible. The most useful one for knowledge representation, because of the deductions that it supports, is t-*entailment*, which is defined as $\alpha \rightarrow_{\mathrm{t}} \beta$ iff for all compatible sets of situations, S, and variable maps, ν, if $\mathrm{t} \in i(\alpha, S, \nu)$ then $\mathrm{t} \in i(\beta, S, \nu)$.

The reason that this logic is of interest for knowledge representation is that t-entailment captures a non-trivial set of deductions and that t-entailment is decidable. The following procedure determines if one sentence in prenex form t-entails another, also in prenex form[8]. First, convert both sentences into conjunctive normal form. Second, Skolemize all existentials in the left hand sentence and all universals in the right hand sentence. Then, if α and β are sentences in prenex conjunctive Skolem form ($\alpha = \forall\vec{z} \bigwedge \alpha_j$ and $\beta = \exists\vec{x} \bigwedge \beta_i$) then $\alpha \rightarrow_{\mathrm{t}} \beta$ iff there exists θ, a substitution for $\vec{x}$, such that for each β_i there exists some α_j and ψ, a substitution for $\vec{z}$, such that $\alpha_j\psi \subseteq \beta_i\theta$ (treating $\alpha_j\psi$ and $\beta_i\theta$ as sets of literals). This theorem about t-entailment can be easily converted into a specification for the algorithm, which demonstrates that t-entailment in this logic is decidable, at least for sentences in prenex form. Although its worst case behavior is exponential in the size of α and β it will always terminate and will be quite fast under normal conditions, where clauses are not long and there are many different predicates.

This variant of first-order tautological entailment is extremely weak. For example, modus ponens is not a valid rule of inference in the logic. Quantification has been weakened so that a universal still t-entails the conjunction of any number of instantiations of itself, but a disjunction of instantiations no longer t-entails an existential. One advantage that this logic has for knowledge representation, along with other relevance logics, is that contradictions do not corrupt the entire reasoning process.

III The Terminological Component

Terminological logics are a formalization of the main ideas of semantic networks and frame-based knowledge representation systems. In general, a terminological logic is a logic of definitions of predicates[9]. The particular logic used here is on a par with the description languages of KL-ONE [4] and NIKL [12], except for its lack of number restrictions, and is much more complicated than the one used in KRYPTON.[10]

The syntax of the logic is as follows:

```
<definition> ::= <1-place-predicate> <rel-op> <concept> |
                 <2-place-predicate> <rel-op> <role>
<rel-op> ::=     = | ≤ | ≥
<concept> ::=    <1-place predicate> |
                 ¬<concept> |
                 (∧ <concept> <concept>) |
                 (∨ <concept> <concept>) |
                 (all <role> <concept>) |
                 (∃ <role>)
<role> ::=       <2-place predicate> |
                 (comp <role> <role>) |
                 (vr <role> <concept>)
```

A property that must be met by any set of definitions in this logic is that the definitions must be unique and must not contain any circular definitions.

The logic is capable of defining concepts such as "a satisfied grandmother is a person all of whose children are doctors or lawyers, all of whose children are married to someone, and who has at least one grandchild". This would be expressed as *satisfied-grandmother* = (∧ *person* (all *child* (∨ *doctor lawyer*)) (all *child* (∃ *spouse*)) (∃ (comp *child child*))).

The semantics makes use of extension functions, $\mathcal{E}^{\mathrm{t}}$ and $\mathcal{E}^{\mathrm{f}}$, which map concepts into subsets of a domain and roles into sets of ordered pairs over the domain. For any concept, c, the set of domain elements in $\mathcal{E}^{\mathrm{t}}[c]$ form the positive extension of c,

[8] The procedure for sentences not in prenex form is more complicated but t-entailment is still decidable for arbitrary sentences. (The proof of this will be in a forthcoming paper.)

[9] The name comes from the fact that these definitions form *terms*.

[10] It is interesting to contrast terminological logics in this context with terminological logics used with standard first-order logic, as in KRYPTON, especially in light of the computational problems reported by Brachman and Levesque [3].

those things in c, and the set of domain elements in $\mathcal{E}^{\mathsf{f}}[c]$ form the negative extension of c, those things not in c. (Because the semantics is four-valued, the positive and negative extensions need not cover the entire domain and also need not be disjoint.) The positive and negative extension of a role are defined similarly, as sets of pairs.

Given a compatible set of situations, S, the extension functions are defined as follows:

$d \in \mathcal{E}^{\mathsf{t}}_S[a]$ iff $\mathsf{t} \in j_s[a](d)$ $\forall s \in S$ for a a 1-place predicate
$d \in \mathcal{E}^{\mathsf{f}}_S[a]$ iff $\mathsf{f} \in j_s[a](d)$ $\forall s \in S$ for a a 1-place predicate
$d \in \mathcal{E}^{\mathsf{t}}_S[\neg c]$ iff $d \in \mathcal{E}^{\mathsf{f}}_S[c]$
$d \in \mathcal{E}^{\mathsf{f}}_S[\neg c]$ iff $d \in \mathcal{E}^{\mathsf{t}}_S[c]$
$d \in \mathcal{E}^{\mathsf{t}}_S[(\wedge\ c_1\ c_2)]$ iff $d \in \mathcal{E}^{\mathsf{t}}_S[c_1]$ & $d \in \mathcal{E}^{\mathsf{t}}_S[c_2]$
$d \in \mathcal{E}^{\mathsf{f}}_S[(\wedge\ c_1\ c_2)]$ iff $\exists S_1 \cup S_2 = S$ $d \in \mathcal{E}^{\mathsf{f}}_{S_1}[c_1]$ & $d \in \mathcal{E}^{\mathsf{f}}_{S_2}[c_2]$
$d \in \mathcal{E}^{\mathsf{t}}_S[(\vee\ c_1\ c_2)]$ iff $\exists S_1 \cup S_2 = S$ $d \in \mathcal{E}^{\mathsf{t}}_{S_1}[c_1]$ & $d \in \mathcal{E}^{\mathsf{t}}_{S_2}[c_2]$
$d \in \mathcal{E}^{\mathsf{f}}_S[(\vee\ c_1\ c_2)]$ iff $d \in \mathcal{E}^{\mathsf{f}}_S[c_1]$ & $d \in \mathcal{E}^{\mathsf{f}}_S[c_2]$
$d \in \mathcal{E}^{\mathsf{t}}_S[(\mathtt{all}\ r\ c)]$ iff
$\forall e \in D \quad \exists S_1 \cup S_2 = S \quad \langle d, e\rangle \in \mathcal{E}^{\mathsf{f}}_{S_1}[r]$ & $e \in \mathcal{E}^{\mathsf{t}}_{S_2}[c]$
$d \in \mathcal{E}^{\mathsf{f}}_S[(\mathtt{all}\ r\ c)]$ iff $\exists e \in D$ $\langle d, e\rangle \in \mathcal{E}^{\mathsf{t}}_S[r]$ & $e \in \mathcal{E}^{\mathsf{f}}_S[c]$
$d \in \mathcal{E}^{\mathsf{t}}_S[(\exists\ r)]$ iff $\exists e \in D$ $\langle d, e\rangle \in \mathcal{E}^{\mathsf{t}}_S[r]$
$d \in \mathcal{E}^{\mathsf{f}}_S[(\exists\ r)]$ iff $\forall e \in D$ $\langle d, e\rangle \in \mathcal{E}^{\mathsf{f}}_S[r]$
$\langle d, e\rangle \in \mathcal{E}^{\mathsf{t}}_S[a]$ iff $\mathsf{t} \in j_s[a](d, e)$ $\forall s \in S$ for a a 2-place pred.
$\langle d, e\rangle \in \mathcal{E}^{\mathsf{f}}_S[a]$ iff $\mathsf{f} \in j_s[a](d, e)$ $\forall s \in S$ for a a 2-place pred.
$\langle d, e\rangle \in \mathcal{E}^{\mathsf{t}}_S[(\mathtt{vr}\ r\ c)]$ iff $\langle d, e\rangle \in \mathcal{E}^{\mathsf{t}}_S[r]$ & $e \in \mathcal{E}^{\mathsf{t}}_S[c]$
$\langle d, e\rangle \in \mathcal{E}^{\mathsf{f}}_S[(\mathtt{vr}\ r\ c)]$ iff
$\exists S_1 \cup S_2 = S \quad \langle d, e\rangle \in \mathcal{E}^{\mathsf{f}}_{S_1}[r]$ & $e \in \mathcal{E}^{\mathsf{f}}_{S_2}[c]$
$\langle d, e\rangle \in \mathcal{E}^{\mathsf{t}}_S[(\mathtt{comp}\ r_1\ r_2)]$ iff
$\exists z \quad \langle d, z\rangle \in \mathcal{E}^{\mathsf{t}}_S[r_1]$ & $\langle z, e\rangle \in \mathcal{E}^{\mathsf{t}}_S[r_2]$
$\langle d, e\rangle \in \mathcal{E}^{\mathsf{f}}_S[(\mathtt{comp}\ r_1\ r_2)]$ iff
$\forall z \quad \exists S_1 \cup S_2 = S \quad \langle d, z\rangle \in \mathcal{E}^{\mathsf{f}}_{S_1}[r_1]$ & $\langle z, e\rangle \in \mathcal{E}^{\mathsf{f}}_{S_2}[r_2]$

A compatible set of situations, S, satisfies the definition $P = T$, precisely when $\mathcal{E}^{\mathsf{t}}_S[P] = \mathcal{E}^{\mathsf{t}}_S[T]$ and $\mathcal{E}^{\mathsf{f}}_S[P] = \mathcal{E}^{\mathsf{f}}_S[T]$, *i.e.*, when the positive and negative extensions of the predicate coincide with the positive and negative extensions of the concept (or role). Also, a compatible set of situations, S, satisfies $P \leq T$ ($P \geq T$), precisely when $\mathcal{E}^{\mathsf{t}}_S[P] \subseteq \mathcal{E}^{\mathsf{t}}_S[T]$ ($\mathcal{E}^{\mathsf{t}}_S[P] \supseteq \mathcal{E}^{\mathsf{t}}_S[T]$) and $\mathcal{E}^{\mathsf{f}}_S[P] \supseteq \mathcal{E}^{\mathsf{f}}_S[T]$ ($\mathcal{E}^{\mathsf{f}}_S[P] \subseteq \mathcal{E}^{\mathsf{f}}_S[T]$), *i.e.*, when the positive and negative extensions of the predicate and the concept (or role) are in correct inclusion relationship.

Terminological logics also have the notion of subsumption between terms. Intuitively, one term subsumes another if the former is more general than the latter. For any two concepts, c and c', c is subsumed by c' under the presence of a set of definitions, $\aleph$, (written $c \overset{\aleph}{\Longrightarrow} c'$) iff for any compatible set of situations, S, such that S satisfies each element of $\aleph$, and for any d in the common domain of the setups, $\mathcal{E}^{\mathsf{t}}_S[c] \subseteq \mathcal{E}^{\mathsf{t}}_S[c']$ and $\mathcal{E}^{\mathsf{f}}_S[c] \supseteq \mathcal{E}^{\mathsf{f}}_S[c']$. That is, one concept is subsumed by a second when all individuals known to be instances of the second must also be known to be instances of the first and all individuals known not to be instances of the first must also be known not to be instances of the second. The similar condition holds for role subsumption.

As expected, subsumption in this logic is weaker than subsumption in standard logics (such as the logic in [3]). For example, (∧ (all *child doctor*) (all (vr *child doctor*) *lawyer*)) ("something all of whose children are doctors and all of whose children which are doctors are lawyers"), is not subsumed by (all *child lawyer*) ("something all of whose children are lawyers"), in this logic, although it would be in a standard logic. The valid subsumptions in this terminological logic are similar to the t-entailments in the decidable variant of first-order tautological entailment described above, and do not include rules like *modus ponens*.[11]

IV The Hybrid Logic

The hybrid logic is now simple to define. The only missing part is the definition of hybrid t-entailment. One formula, α, t-entails another, β, in the presence of a set of definitions, $\aleph$, written $\alpha \xrightarrow{\aleph} \beta$, iff for all compatible sets of situations, S, such that S satisfies each element of $\aleph$, and for all variable maps, ν, if $\mathsf{t} \in i(\alpha, S, \nu)$ then $\mathsf{t} \in i(\beta, S, \nu)$.

This hybrid logic is deductively more powerful than its assertional component. The assertional component cannot draw the conclusion that Tweety is an animal from the facts that Tweety is a bird and that all birds are animals. This deduction can be performed in the hybrid logic by using the definition $bird \leq animal$. In the presence of this definition, $bird(Tweety)$ t-entails $animal(Tweety)$.

Note, however, that the definition $bird \leq animal$ is different expressively than the assertion $\forall x\ \neg bird(x) \vee animal(x)$. Definitions in terminological hybrids are different than universally quantified statements, in that a definition defines the intensional meaning of a predicate, whereas a universally quantified statement is simply a conditional statement about the current state of the world.

It is not an accident that the definitions of the extension functions are so similar to the definitions of the support relationships. This was done for two reasons. First, determining t-entailment in the hybrid logic can be done by taking any occurrences of defined predicates and replacing them by formulae using only primitive predicates. This implements hybrid t-entailment in non-hybrid t-entailment. Second, determining subsumption in the terminological logic can be performed by mechanically translating the terms into formulae in the assertional logic and testing to see if the first t-entails the second, under the same set of definitions. Therefore the algorithm for determining t-entailment in the non-hybrid language can be used for both of these problems, thus demonstrating that they are decidable.[12]

V Knowledge Representation System Specification

The above section gave a description of a hybrid, decidable logic. However, this is not a quite specification of a knowledge representation system based on the logic. A knowledge representation system interacts with its users by means of a set of operations and these operations remain to be defined.[13]

Underlying the definitions of the operations given here is the notion of a knowledge base. A knowledge base is a pair, $\langle \mathcal{D}, \mathcal{S}\rangle$, where $\mathcal{D}$ is a set of definitions and $\mathcal{S}$ is a set of sentences. The idea behind a knowledge base (as will be obvious in the specifications of the operations) is that each definition and each sentence in the knowledge base are taken to be true. The operations allow adding new definitions and sentences to the knowledge base, and answering questions about what follows from the knowledge base. All of these operations have simple meanings in the semantics of the hybrid, decidable logic. Together, they provide a full set of

[11] Interestingly, subsumption in the logic is very close to what is actually computed by the classification algorithm of NIKL.

[12] A proof of these results will be in a forthcoming paper.

[13] A similar definition of KRYPTON is given in [5].

capabilities for the knowledge representation system. No other operations are allowed on the knowledge base.

The first operation simply generates the empty knowledge base:

$$\text{NEWKB}[] = \langle\{\}, \{\}\rangle$$

There are two operations that add information to the knowledge base, one to add definitions and one to add assertions:

$$\text{DEFINE}[\beta, \langle \mathcal{D}, \mathcal{S}\rangle] = \langle\{\beta\} \cup \mathcal{D}, \mathcal{S}\rangle$$
$$\text{TELL}[\alpha, \langle \mathcal{D}, \mathcal{S}\rangle] = \langle \mathcal{D}, \mathcal{S} \cup \{\alpha\}\rangle$$

where α is a sentence and β is a definition. Finally, there are three operations that query the knowledge base. The first two of these are:

$$\text{SUBSUMES}[c_1, c_2, \langle \mathcal{D}, \mathcal{S}\rangle] = \begin{cases} \text{yes, if } c_1 \overset{\mathcal{D}}{\Longrightarrow} c_2 \\ \text{no, otherwise} \end{cases}$$

$$\text{ASK}[\alpha, \langle \mathcal{D}, \mathcal{S}\rangle] = \begin{cases} \text{yes, if } \bigwedge \mathcal{S} \overset{\mathcal{D}}{\longrightarrow} \alpha \\ \text{no, otherwise} \end{cases}$$

where α is a sentence and c_1 and c_2 are concepts or roles. The first operation thus determines whether one concept (or role) subsumes another, given the current set of definitions, and the second determines whether some formula follows from the current knowledge base. The last query operation is SHOW, which is like ASK except that it returns the possible bindings for the existentially quantified variables of α.

These definitions, plus the facts derived above about the logic, lead directly to an algorithmic characterization of the knowledge representation system. The non-housekeeping portion of the knowledge representation system will be the algorithm for t-entailment presented above.

This is not to say that the development of the knowledge representation system would be easy as the t-entailment algorithm is not trivial. One improvement would be to modify the specification from one with an explicit knowledge base into one with a single, implicit knowledge base. Also, the exact representation of the knowledge base would make a considerable difference on the speed of the system. Finally, the t-entailment algorithm is highly parallel and would benefit from parallel implementation.

VI Summary

The above development shows how a hybrid, decidable, logic-based knowledge representation system can be built. This knowledge representation system has several desirable properties. First, it is truly logic-based. The behavior of the system is completely determined by a specification that directly refers to a reasonable logic. Second, all of its operations are decidable, so that all operations will always terminate with results defined by the logic. Third, it is based on a hybrid logic that is more powerful than its assertional component. This extra power comes in an area that is very important for AI. Fourth, the main algorithms of the system are highly parallel, indicating that the system could be usefully implemented on a massively parallel machine, especially if the knowledge base was very large.

There are, however, some undesirable properties of the system. Both components of the logic are still very weak, leading to a system with few inferences. This seems to be an unavoidable characteristic of any general-purpose decidable first-order logic. To use the system for any real task, it would have to be just a part of larger reasoning system which treats the system as a sound but not complete reasoner. Also, some operations are NP-complete in the size of the query so its worst-case performance will be very slow. There seems to be no way around this if any interesting inferences are to be done with quantifiers, since theta-subsumption (a main part of the t-entailment algorithm and a seemingly basic part of any interesting quantificational inferences) is itself NP-complete.

One possible extension of the knowledge representation system is to create a hybrid logic with many auxiliary logics. These auxiliary logics would be domain dependent and decidable, but of reasonable power. In this way a more powerful knowledge representation system could be built, one that might be able to solve interesting problems in its own right and still be guaranteed to always terminate.

VII Acknowledgments

Hector Levesque first introduced me to tautological entailments. His assistance and encouragement is greatly appreciated. Ron Brachman, as the past leader of the knowledge representation group at SPAR, was instrumental in maintaining a comfortable research environment. Pat Hayes has capably stepped into this role and also gave voluminous comments on an earlier draft of this paper.

References

[1] Alan Ross Anderson and Nuel D. Belnap, Jr. *Entailment: The Logic of Relevance and Necessity*. Volume I, Princeton University Press, Princeton, New Jersey, 1975.

[2] Nuel D. Belnap, Jr. A useful four-valued logic. In G. Epstein and J. M. Dunn, editors, *Modern Uses of Multiple-Valued Logic*, pages 8–37, Reidel, 1977.

[3] Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings AAAI-84*, pages 34–37, August 1984.

[4] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, April-June 1985.

[5] Ronald J. Brachman, Victoria Pigman Gilbert, and Hector J. Levesque. An essential hybrid reasoning system: knowledge and symbol level accounts of KRYPTON. In *Proceedings IJCAI-85*, pages 532–539, August 1985.

[6] Alan M. Frisch. Using model theory to specify AI programs. In *Proceedings IJCAI-85*, pages 148–154, August 1985.

[7] Patrick J. Hayes. In defence of logic. In *Proceedings IJCAI-77*, pages 559–565, August 1977.

[8] Hector J. Levesque. A logic of implicit and explicit belief. In *Proceedings AAAI-84*, pages 198–202, August 1984.

[9] João P. Martins and Stuart C. Shapiro. A model for belief revision. In *Proceedings of the Non-Monotonic Reasoning Workshop*, pages 241–294, October 1984.

[10] Robert C. Moore. The role of logic in knowledge representation and commonsense reasoning. In *Proceedings AAAI-82*, pages 428–433, August 1982.

[11] Peter F. Patel-Schneider. A decidable first-order logic for knowledge representation. In *Proceedings IJCAI-85*, pages 455–458, August 1985.

[12] James G. Schmolze. The language and semantics of NIKL. Draft, BBN Laboratories, April 1985.

[13] Stuart C. Shapiro and Mitchell Wand. *The Relevance of Relevance*. Technical Report 46, Computer Science Department, Indiana University, November 1976.

The Generalized-Concept Formalism - a Frames and Logic Based Representation Model

Mira Balaban

Department of Computer Science
State University of New York at Albany
1400 Washington Avenue, Albany, NY 12222

Abstract

A representation formalism that is both frame-based and logic-based is defined. The formalism, entitled, "The Generalized-Concept (G-C) Model", allows for the naturality of structuring knowledge around objects of the subject domain, and enjoys the rigorousness of an underlying logic language. Properties of the formalism are discussed, and compared with similar properties of frames systems. It is claimed that the G-C model extends the common frames notation, while avoiding some of the characteristic problems of frames. The G-C model was developed within a research on the establishment of a formal basis for the study of Western Tonal Music (WTM). It is currently used in a system that is intended to support a general study of WTM.

1. Introduction

Frame based systems gained a widespread popularity in AI and motivated the development of knowledge representation languages like KRL [5], and KL-ONE [8]. Their popularity results, probably, from the naturality of organizing the knowledge as taxonomies of generic frames and their instances. The main disadvantage of frames formalisms is the lack of formal semantics, and sometimes, the peculiarities of their syntax. Logic languages provide the simple and rigorous syntax and semantics that frames systems lack, but lack the intuitive appeal and the relative efficiency of frames formalisms. This situation stimulated work on hybrid systems, i.e., systems that employ multiple representation formalisms ([11], [15], [16], [18], [19], [9], [10]).

The Generalized-Concept (G-C) formalism combines the conceptual approach of frames formalisms with the rigorousness and simplicity of logic. A G-C representation associates objects in the problem domain with their descriptions (in logic) and is obtained by successive refinements of a frames like conceptual description. Additional naturality is provided by a frames like interface language, that abbreviates logic constructs. We think that the main contribution of the formalism is for the representation of large bodies of knowledge in complex domains, where structuring of knowledge and rigorousness of description are essential for reliability.

The formalism was developed for, and is currently used in, a system that is intended to support a general, formal study of Western Tonal Music (WTM) ([2]-[4]). In this paper we describe the formalism and discuss its advantages over common frames formalisms. The examples are all taken from the music domain.

2. The G-C Formalism in Brief

A representation language must have simple and rigorous syntax and semantics. But for large and complex domains it should also provide help in, probably, the most difficult step of any representation process, i.e., the translation of real world knowledge into formal representation. Without that, the results might be dubious, in spite of the formal semantics. From this point of view logic languages are low level representation languages. The transformation of conceptually structured knowledge into rigorous descriptions is the essence of the G-C formalism.

We assume that the knowledge about the subject domain consists of a *characteristic part*, called here a *theory* of the subject domain, and an *anecdotal part*. The theory represents the stable part of the problem. It describes "typical" objects in the relevant world. The anecdotal part represents the changing part of the problem. It describes specific knowledge about objects described in the theory.

A theory of the relevant world is transformed, by two successive abstractions, into a *G-C Knowledge Base (GCKB)*, which is a set of constraints-systems, chunked according to the objects described by the theory. A GCKB is given a fixpoint semantics (or model-theoretic semantics). However, the meaning of elements of a GCKB can be explained in an intuitive way, so that a potential user does not have to be "expert" in fixpoint or model theories[a]. The anecdotal knowledge would be formally represented in the constraints language of the GCKB. A GCKB, augmented with formally represented anecdotal knowledge, is a *Complete GCKB*.

The classification of knowledge into theory and anecdotal parts reminds the terminological and assertional components of KRYPTON ([9], [10]), and the quantificational and propositional reasoners of KL-TWO ([19]). The main differences are: 1) Our classification is based on the role of the described knowledge (stable versus changing) and not on the kinds of objects (descriptions versus statements) as in KRYPTON or on the representation language (quantificational versus propositional) as in KL-TWO. 2) Knowledge of the real world is modelled in a different way. 3) A GCKB is assigned global semantics that defines the conceptual theories that it describes. 4) No interface between the theory and the anecdotal parts is necessary since their formal representation uses the same constraints language.

3. From theory to GCKB

A G-C representation is developed in three stages. On the first stage, the objects of the conceptual theory and their inter-relations are determined. On the second stage the conceptual theory is transformed into a uniform set of units called Generalized-Concepts (G-C-s). On the last formal stage, G-C's are modelled by constraints systems. The levels construction yields a formal description that is structured according to the conceptual objects.

---

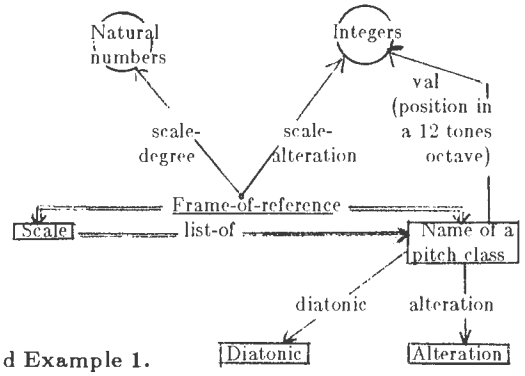
[a] Like a PROLOG user that does not, necessarily, have to know resolution proof theory, or a LISP user that can know nothing about λ-calculus and fixpoint semantics.

3.1 Conceptual framework

We assume two kinds of conceptual objects: *concepts* and *relationships*. Concepts represent collections of entities that have similar content. The information content of a concept is given by a finite number of *attributes*, which are mappings from, or relations between, the concept to other domains. Each attribute is associated with one domain. Relationships are relations among concepts, or relations on a set of concepts. They also may have attributes.

The main examples of this paper deal with the transformation of a fairly complicated musical relationship into its G-C representation. We now describe the musical knowledge that is necessary in order to understand the examples. A *name-of-a-pitch-class* is a pair of a *diatonic-name* (A-G) and an *alteration* (one of: bb,b,♮,#,##). Examples are: (G,#), (C,b), (F,##). The fundamental order of the diatonic-names is A-G. This is a cyclic order (A follows G). Each name-of-a-pitch-class denotes a tone in a twelve tones octave. This is denoted by the attribute *val*. A *scale* is a list of name-of-pitch-classes that their diatonic-names are ordered, and every diatonic-name appears exactly once in the scale. An example of a scale (the C# major scale) is: ((C,#), (D,#), (E,#), (F,#), (G,#), (A,#), (B,#)). The Scale concept is a frame–of–reference to the Name-of-a-pitch-class concept. This relationship has two attributes: *scale-degree* and *scale-alteration*. The scale-degree and scale-alteration of a name-of-a-pitch-class (d,a) with respect to a scale s, are, respectively, the position number of the (unique) element of s that has d as its diatonic-name, and the difference between the values of the val attribute of (d,a) and that element. For example, the scale-degree of (G,b) in the C# major scale (above) is 5; its alteration is -2.

Example 1 – Conceptual description of the frame–of–reference relationship between the Name-of-a-pitch-Class and the Scale concepts: We use graphical notation: Boxes denote concepts; circles denote attribute-domains that are not concepts or relationships; labeled arcs represent attributes; labeled double arcs represent relationships involving the concepts at the arc ends. Attributes of the related concepts that are relevant for later modeling of this relationship (see examples below) are also shown.



End Example 1.

The "finite-representation" assumption: Except for a small number of "base objects", the instances and attribute value assignments of a concept or a relationship admit **regularity** that can be **finitely described**. The description may involve other concepts and relationships and their attributes.

Base objects are either: *Finite* or *Primitive*. Primitive objects are those that are not finite, and cannot be finitely described because such a characterization is not known. (Non-analytic concepts, see [8]). For example, the concept of a Tonal-Music-Piece is primitive. We assume that each primitive object has an associated *typing predicate*. Clearly, this just throws the bulk of the primitivity from the object to the typing predicate, but it is useful here since it leaves us with a clean conceptual framework, in which the finite-representation assumption applies to **all** objects of the subject domain. The "typing predicate" assumption can be removed, later on, by the abbreviation language.

3.2 Generalized-Concepts (G-C-s) Level

This level is the bridge between the informal conceptual level to the formal GCKB level. It has a double role: to clean and unify the conceptual description. To clean means to get rid of overlapping descriptions. For example, the *list-of* relationship between te Scale and Name-of-a-pitch-class concepts in example 1 is redundant since this information would be part of the definition of the Scale concept. The "unify" role is the organization of the conceptual information as a set of uniform objects called *Generalized-Concepts (G-C-s)*. G-C-s are defined as follows: The content of an object is considered as a *main* attribute, mapping the object's instances into their contents. An object "c" that has the attributes $\{a_0,a_1,...,a_n\}$ $(n \geq 0)$, where a_0, denotes the main attribute and any attribute a_i $(0 \leq i \leq n)$ has a domain $D(a_i)$, is represented by a G-C "C", defined as follows: 1) $C \subseteq D(a_0) \times D(a_1)... \times D(a_n)$; 2) Every tuple in "C" describes a combination of attributes' values (including the main attribute) assigned to an entity in "C".

Example 2 – A G-C representation for the frame–of–reference relationship between the Name-of-a-pitch-class and the Scale concepts: The relationship is modelled by a G-C named PITCH-CLASS-NAME-OVER-A-SCALE, which is the set of all tuples (*pitch-class-name, diatonic, alteration, val, scale, scale-degree, scale-alteration*), such that (*pitch-class-name, diatonic, alteration, val*) is a tuple in the PITCH-CLASS-NAME G-C (the G-C that models the Name-of-a-pitch-class concept); *scale* is the first element in a tuple of the SCALE G-C (the G-C that models the scale concept); *scale-degree* is the degree of *pitch-class-name* in *scale*, and *scale-alteration* is the alteration of *pitch-class-name* in *scale*.

End Example 2.

The result of this level, is a set of G-C-s called a *G-C theory*.

3.3 G-C Knowledge Base

This level is based on the finite-representation assumption. A G-C is modeled by a system of constraints called a *Definition-of-a-Generalized-Concept (D-G-C)*. The constraints language is assumed to be some variant of a many sorted first-order logic language. A set of D-G-C-s, together with the theories of the underlying domains (lists, arithmetics, sets, etc.), forms a *G-C Knowledge Base (GCKB)*, whose denotations, in terms of fixpoints, are G-C Theories. The need for fixpoint semantics is not surprising, since, recursion appears in very basic concepts, like, for example, the Twelve-Tones-String concept in the music domain.

3.3.1 Syntax of a GCKB

The D-G-C-s language is based on the distinction of three special kinds of symbols: 1) Concept symbols: $C_i^{n_i}$ $(1 \leq i \leq k, n_i \geq 1$, for some $k \geq 1)$, that are used as n_i–ary predicate symbols (printed in italic upper case letters); 2) main-attribute symbols: $a_0^{(i)}(1 \leq i \leq k)$, and attribute symbols: $a_j^{(i)}(1 \leq i \leq k, 1 \leq j \leq n_{i-1})$. Both are used as individual variables (printed in italic lower case letters). In addition, the language might include any symbol of a many sorted logic languages. A *D-G-C* for a G-C "C_i" $(1 \leq i \leq k)$, is a wff of the form

$$C_i^{n_i}(a_0^{(i)}, a_1^{(i)}, ..., a_{n_{i-1}}^{(i)}) \equiv \mathbf{F}[C_1^{n_1}, ..., C_k^{n_k}](a_0^{(i)}, ..., a_{n_{i-1}}^{(i)})$$

where **F** denotes the set of constraints that define the G-C "C_i." **F** might have the concept symbols for the rest of the G-C-s, and the main-attribute and attribute symbols for "C_i" as variables. **F** is called the *defining expression* of the D-G-C. The intuitive meaning of the D-G-C is that a given instantiation of the a_i -s is an element of "C_i" iff it satisfies **F**. Denote this D-G-C by $\mathbf{C_i}$ and let **DK** denote the theories of the underlying domains, then, $\{\mathbf{C_i}\}_{i=1}^{k} \cup \mathbf{DK}$ is a GCKB.

Example 3 – A D-G-C modeling the PITCH-CLASS-NAME-OVER-A-SCALE G-C: The constraints language is first order predicate calculus, with a unique sort. The Concept symbols are *PITCH-CLASS-NAME-OVER-A-SCALE, PITCH-CLASS-NAME, SCALE;* the main-attribute symbol is *pitch-class-name;* the attribute symbols are *diatonic, alteration, val, scale, scale-degree, scale-alteration.*

We use the following abbreviations: 1) A constraint that says that $e_{i_1},...,e_{i_l}$ satisfy the D-G-C for the C_j G-C when e_{i_m} is substituted for $a_{i_m}^{(j)}$ $(1 \le m \le l)$, should have the format:

$$\exists \overline{s}\ C_j(a_0^{(j)}, a_1^{(j)}, ..., a_{n_j}^{(j)})\, s$$

where s is the substitution $s = \{e_{i_1}/a_{i_1}^{(j)}, ..., e_{i_l}/a_{i_l}^{(j)}\}$, and $\overline{s}$ includes all $a_i^{(j)}$-s not appearing in s. We abbreviate this by the "frames like" notation

C_j with
$a_{i_1}^{(j)}$: e_{i_1}
⋮
$a_{i_l}^{(j)}$: e_{i_l}

That is, a pair like "$a_i^{(j)}$: e_i" stands for "the attribute symbol $a_i^{(j)}$ is substituted by e_i, and unspecified attribute symbols are assumed to be existentially quantified. This convention saves the need to know the main-attribute and all attribute symbols of a D-G-C, including their order. It relies on the distinction of main-attribute and attribute symbols from the other individual variables. 2) Attribute symbols in different D-G-C-s might have common names.

There are three constraints: **A1-A3.** Constraints **A1** and **A2** are "is-a" constraints, defining the relevant attributes of Name-of-a-pitch-class and Scale; the third constraint defines the relationship's attributes.

A1. *pitch-class-name, diatonic, alteration,* and *val* form an instance of the PITCH-CLASS-NAME G-C:

PITCH-CLASS-NAME with
pitch-class-name: pitch-class-name.
diatonic: diatonic.
alteration: alteration.
val: val.

A2. *scale* is a scale, i.e., there is a tuple in the SCALE G-C that has *scale* in its main-attribute position:

SCALE with
scale: scale.

A3. Definition of *scale-degree* and *scale-alteration* as given in the introductory paragraph to example 1.

PITCH-CLASS-NAME with
pitch-class-name: ELT(*scale-degree, scale).*
diatonic: diatonic.
val: val – scale-alteration).

Note that *scale-degree* and *scale-alteration* are **implicitly** defined.

The full D-G-C is:

PITCH-CLASS-NAME-OVER-A-SCALE (pitch-class-name, diatonic, alteration, val, scale, scale-degree, scale-alteration) ≡ A1 ∧ A2 ∧ A3.

Example instance: The following values satisfy the D-G-C for the NOTE-NAME-OVER-A-SCALE G-C:

note-name = (E,b), *diatonic* = E, *alteration* = b, *val* = 3,
scale = ((C,#),(D,#),(E,#),(F,#),(G,#),(A,#),(B,#)) (i.e., C# major),
scale-degree = 3, *scale-alteration* = 2.

A GCKB may be obtained by providing D-G-C-s for PITCH-CLASS-NAME, SCALE, and all the G-C-s they rely on, and providing the theories of the underlying domains.

End Example 3.

3.3.2 Semantics of a GCKB

A fixpoint semantics to a GCKB $\mathbf{G} = \{\mathbf{C_i}\}_{i=1}^{k} \cup \mathbf{DK}$ is defined as follows: Let I be an interpretation of the sort symbols and the individual/function/predicate symbols with respect to their sorts, such that I satisfies **DK**. Denote the interpreted **G** by $\mathbf{G}^I$. Note that concept, main-attribute, and attribute symbols are left uninterpreted, but with associated sorts. Assign to a concept symbol C_i , a predicate P_i, whose domain is "compatible" with the sortal information assigned to C_i in I. Denote by I' the interpretation I augmented with these assignments. Let DE_i, be the defining expression of $\mathbf{C_i}$. Then $DE_i^{I'}$ denotes an n_i-ary predicate over the cartesian product of the domains assigned to $a_j^{(i)}$ $(0 \le j \le n_{i-1})$. We define mappings as follows:

$$T_i^I((P_i)_{i=1}^{k}) = DE_i^{I'} \qquad (1 \le i \le k)$$

that is, T_i^I assigns to $(P_i)_{i=1}^{k}$ the predicate obtained from DE_i by the interpretation I'. The mapping, in I of the GCKB **G**, is defined to be:

$$T_{\mathbf{G}}^I((P_i)_{i=1}^{k}) = (T_i^I((P_i)_{i=1}^{k}))_{i=1}^{k}$$

For a G-C C, denote by P_C its "characteristic predicate", i.e., a predicate on the domain of C that is true exactly on C. Then, a G-C theory $\{C_i\}_{i=1}^{k}$ is said to be *defined by,* the GCKB **G**, if there exists an interpretation I, such that $(P_{C_i})_{i=1}^{k}$ is a fixpoint of $T_{\mathbf{G}}^I$, i.e.,

$$T_{\mathbf{G}}^I((P_{C_i})_{i=1}^{k}) = (P_{C_i})_{i=1}^{k}$$

An equivalent model theoretic semantics can be defined as follows: Close all D-G-C-s in **G** by universal quantification. Denote the result by $\overline{\mathbf{G}}$. Let I be a model for $\overline{\mathbf{G}}$. Then the extensions of the C_i -s $(1 \le i \le k)$ in I, form a G-C theory that is defined by **G** through I.

4. Complete GCKB

A *complete GCKB* is a pair **(G,A)**, where **G** is a GCKB and **A** is a set of wff-s in the constraints language of **G**. **A** is called the *anecdotal component.* A G-C theory *is defined by* the complete GCKB **(G,A)** if it is defined by **G** through a model I of **G**, that is also a model of **A**.

Example 4 – Anecdotal knowledge:

Assume that we are uncertain about the exact melody of the tune named "Twinkle twinkle little star". We have two candidate melodies, denoted by Melody1 and Melody2, and both are written in a major scale. This knowledge would be included in the anecdotal part as follows:

MELODY-OVER-A-SCALE with
melody: Melody1
name: TWINKLE-TWINKLE
scale-kind: MAJOR

∨

MELODY-OVER-A-SCALE with
melody: Melody2
name: TWINKLE-TWINKLE
scale-kind: MAJOR

5. Application

The expected usage of a complete GCKB is to answer queries about objects described in the theory of the subject domain. Queries might be of three basic kinds: *prove, find* (prove/find an element of a G-C), and *partial prove/find* (complete a partially defined tuple into an element of a G-C). Basic kind queries might be combined to form complex queries. Answering queries requires an agreement on a unique G-C theory defined by a complete GCKB, and Procedural Interpretation for the unique meaning.

Unique meaning: The denotation of a complete GCKB **(G,A)** can be taken as the G-C theory that corresponds to the least fixpoint in the standard interpretation of **G**, provided that it exists and satisfies the standardly interpreted anecdotal component. Note, however, that the "optimal" G-C theory defined by a complete GCKB is, as a matter of fact, the greatest G-C theory obtained from it – provided that it exists[(b)]. In the model-theoretic semantics, a unique meaning, that might correspond to the least fixpoint semantics, would be the extensions of the concept symbols in the least Herbrand model of the complete GCKB, provided that it exists. The problem, in both cases, is the existence of the least fixpoint or Herbrand model[(c)]. As a compromise, we suggest to replace our global approach to meaning – What is the G-C theory defined by a complete GCKB, by a local one – what instances of a G-C are implied from a complete GCKB. In the local approach, the denotation of a concept symbol $C_i^{n_i}$ in a complete GCKB **(G,A)**, can be defined by:

$$D_i = \{ (t_1,...,t_{n_i}) \ / \ (\overline{\mathbf{G}},\mathbf{A}) \models C_i^{n_i} (t_1,...,t_{n_i}) \}$$

where $\overline{\mathbf{G}}$ is **G** with universally closed D-G-C-s. We expect that if the least Herbrand model/fixpoint exist, then $\{ D_i \}_{i=1}^{k}$ would be the unique G-C theory defined by **(G,A)**. However, in the general case, $\{ D_i \}_{i=1}^{k}$ is not necessarily a G-C theory defined by **(G,A)**. D_i can be understood as the set of elements on which all the G-C-s that can be defined by $\mathbf{C}_i$ agree.

Procedural interpretation: Under the local approach to meaning, answers to basic kind queries for a complete GCKB **(G,A)**, can be obtained by any *constructive* theorem prover, i.e., a theorem prover that in the course of a proof produces bindings to existentially quantified attribute symbols. For a first order constraints language, resolution proof theory ([17]) has exactly this constructiveness property. The completeness of the resolution refutation method guarantees that every element in the denotation of a concept symbol can be proved or found.

6. Reasoning with a complete GCKB

Internally, a complete GCKB is just a set of logic formulas, and therefore the inference mechanism that answers these queries can be any "constructive" theorem prover. For example, if the GCKB includes a D-G-C for the MELODY-OVER-A-SCALE G-C, and for all the D-G-C-s that it depends on, and the anecdotal part is that of example 4, then we can prove queries like "There exists a melody by the name of 'twinkle-twinkle'". This would be formulated, using the abbreviation language, as

MELODY with
name: TWINKLE-TWINKLE

The proof relies on an "is-a" constraint of the form

MELODY with
melody: *melody*
name: *name*

in the D-G-C for the MELODY-OVER-A-SCALE G.C.

The resemblance of D-G-C-s to clauses of a logic program suggests the idea of a logic programming system like PROLOG as an inference mechanism. It is expected also that the logic programming specification of the theories of the underlying domains would be fairly simple.

In the music system, PROLOG is indeed used as the inference mechanism. The translation of D-G-C-s into PROLOG was done by hand and required deep understanding of both the D-G-C-s and PROLOG. A special difficulty results from the absolute nondirectional nature of the task. That is, prove, find, and partial prove/find queries, all should be handled by the same mechanism. A special, explicit control strategy (see [12]) developed to suppress PROLOG's static control whenever needed, partially solves this problem. By now, the music system can reason about notes and intervals and their arithmetics. Current work concentrates on the recursive concept of Twelve-Tone-Strings.

7. The G-C formalism as a frames formalism

A D-G-C

$$C_i(a_0^{(i)},...,a_{n_{i-1}}^{(i)}) \equiv \mathbf{F}\,[C_1,...,C_k]\,(a_0^{(i)},...,a_{n_{i-1}}^{(i)})$$

can be viewed as a frame

C_i :
$a_0^{(i)}$: $a_0^{(i)}$
:
$a_{n_{i-1}}^{(i)}$: $a_{n_{i-1}}^{(i)}$
constraints: $\mathbf{F}\,[C_1,...,C_k](a_0^{(i)},...,a_{n_{i-1}}^{(i)})$.

where C_i is the name of the frame; "$a_0^{(i)},...,a_{n_{i-1}}^{(i)}$" and "constraints" are the slot names, and $a_0^{(i)},...,a_{n_{i-1}}^{(i)}$ are variable symbols. Under this view, a GCKB is a system of frames augmented with some Domain Knowledge. Its main advantage over common frames languages is that it is a **clean and powerful frames language**, with simple and rigorous syntax and semantics. In particular, there is no need for special constructs like those of KRL and KL-ONE, and a frames like abbreviations language can provide all the conveniences of frames languages. Below we discuss some important properties of the G-C formalism:

1. controlled property inheritance: There is no built-in provision for complete property inheritance. G-C-s can inherit **any combination of properties** from other G-C-s. Consider, for example, the idea of a metaphor (see [13]). The problem is how to represent a "pig view" of a man, while we don't mean that "a man is a pig", since this would imply a complete inheritance of all of the pig's properties. In frames formalisms, where property inheritance is built into the reasoning procedures, representation of metaphors requires a mechanism for cancellation of properties, which causes some fundamental semantical problems (see [6], [7]). In the G-C formalism, inheritance of any of the properties of another G-C is possible, just by means of attributes substitution. For example, the pig view of a man can be expressed by a "metaphor" constraint of the form

(b) We would like to remove the requirement for standard interpretation by switching to interpretations over the Herbrand Universe of the complete GCKB. In [20] and [1] a fixpoint semantics for a Horn clause program is defined over the Herbrand Universe of the program. We don't know yet if this definition can be extended to complete GCKB-s.

(c) It has been shown, in [20], that for Horn clause programs, both exist and are equal.

PIG with

snout : nose

sty : home

legs : legs

character : character

in the D-G-C for the Pig-like-man concept. The difference between a "metaphor" constraint to an "is-a" constraint (like constraints **A1** and **A2** of example 3) is that in the latter, the main attribute of the embedded D-G-C is replaced. The abbreviation language can be used to save the need to explicitly specify all inherited properties in case of complete property inheritance.

2. A D-G-C provides an **implicit** definition for attributes' values since there is no requirement for explicit attribute-value pairings. We found this very useful in providing natural description for implicitly defined attributes, like the *scale-degree* and *scale-alteration* attributes in Example 3.

3. Composite descriptions: The substitution mechanism of first order logic allows the generation of composite descriptions, without having to define composite D-G-C-s. For example, one can refer to "A Mozart melody in a Minor scale that starts and ends in the same note" just by having D-G-C-s to "A Mozart piece", "A melody over a scale" and "A Minor scale". There is no need to produce first a D-G-C for this composite object. (This property results from viewing slot names as individual variables rather than, according to the more common view, as predicate symbols.)

4. Extending the set of constrains in a D-G-C: The uniform structure of D-G-C-s enables us to refer to restrictions like "a melody of length less than eight", without having to define special D-G-C-s. This can be considered as an abbreviation feature.

5. Default reasoning: The G-C formalism does not provide any special reasoning features like default reasoning or disjointness reasoning. These features should be part of the constraints language or the inference mechanism (like in [14] or in PROLOG). Some default reasoning power is obtained when information is encoded in the attributes instead of in the constraints.

8. Conclusion

We have described a formalism that combines a logic language with structuring of knowledge as in frames representations. The formalism can be viewed as a powerful frames formalism, that can express both descriptional and assertional knowledge, is natural, and has rigorous semantics. Two special properties are the controlled property-inheritance and the implicit definition of attributes. We have claimed that these properties extend the power of common frames languages and eliminate the need for cancellations of attributes (thus avoiding problems of meaning when cancellation is allowed). The steps in the development of a GCKB simplify the process of generating a formal representation to a body of knowledge.

On the practical level, the formalism should be tried in non-musical domains, and there is a need for an abbreviation language to support for example, complete property inheritance, and "is a" and "metaphor" constraints. On the theoretical level, we would like to have better understanding of the semantics of a complete GCKB, and especially of the relation between the global and the local approaches to its meaning. Another interesting subject is the possibility of an automatic translation of a complete GCKB into a logic program.

References

[1] Apt, K.R., and Van Emden M.H., "Contributions to the Theory of Logic Programming" **JACM,** 29(3), 1982, pp. 841-862.

[2] Balaban, M., **"Towards a Computerized Analytical Research of Tonal Music."** Ph.D. Dissertation, The Weizmann Institute of Science, Rehovot, Israel, 1982.

[3] Balaban, M., "CSM - An AI Approach to the Study of Western Tonal Music" **Intelligent Systems and Machines,** Okland University, Mich., 1985.

[4] Balaban, M., "A Formal Basis for Research in Theories of Western Tonal Music - an Artificial Intelligence Approach" **Communication and Cognition, Special Issue on Music and AI,** 1986, (to appear).

[5] Bobrow, D.G., and Winograd, T., "An Overview of KRL, A Knowledge Representation Language", **Cognitive Science,** 1(1), 1977, pp. 3-46.

[6] Brachman, R.J., "What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks" **Computer,** 16(10), 1982, pp. 30-36.

[7] Brachman, R.J., " 'I Lied About the Trees', or, Defaults and Definitions in Knowledge Representation" **The AI Magazine,** 5(3), 1985, pp. 80-93.

[8] Brachman, R.J., and Schmolze, J.G., "An Overview of the KL-ONE Knowledge Representation System", **Cognitive Science,** 9, 1985, pp. 171-216.

[9] Brachman, R.J., Fikes, R.E., and Levesque, J.L., "KRYPTON: A Functional Approach to Knowledge Representation", **Computer,** 16(10), 1983, pp. 67-73.

[10] Brachman, R.J., Gilbert, V.P., and Levesque, J.L., "An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of KRYPTON", **IJCAI,** 1985, pp. 532-539.

[11] Charniak, E., "A Common Representation for Problem Solving and Language Comprehension Information", **Artificial Intelligence,** 16,3, 1981, pp. 225-255.

[12] Gallaire, H. and Lasserre, C., "Metalevel Control for Logic Programs", In **Logic Programming,** K.L. Clark and S.-A. Tarnlund, Eds., Academic Press, 1983, pp. 173-185.

[13] Hays, P.J., "The Logic of Frames", in **Frame Conceptions and Text Understanding,** Metzing, D., (ed), Walter de Gruyter and Co., Berlin, 1979, pp. 46-61.

[14] Reiter, R., "A Logic for Default Reasoning" **Artificial Intelligence** 13(1,2), 1980, pp. 81-132.

[15] Rich, C., "Knowledge Representation Languages and Predicate Calculus: How to Have Your Cake and Eat It Too", **AAAI,** 1982, pp. 192-196.

[16] Rich, C., "The Layered Architecture of a System for Reasoning About Programs" **IJCAI,** 1985, pp.540-546.

[17] Robinson, J.A., "A Machine-Oriented Logic Based on the Resolution Principle", **JACM,** 12(1), 1965, pp. 23-44.

[18] Schubert, L.K., Papalaskaris, M.A., and Taugher, J., "Determining Type, Part, Color, and Time Relationships", **Computer,** 16(10), 1983, pp. 53-60.

[19] Vilain, M., "The Restricted Language Architecture of a Hybrid Representation System" **IJCAI,** 1985, pp. 547-551.

[20] Van Emden, M.H., and Kowalski, R.A., "The Semantics of Predicate Logic as a Programming Language" **JACM,** 23(4), 1976, pp. 733-742.

Knowledge modules vs knowledge-bases: a structure for representing the granularity of real-world knowledge

Diego Lo Giudice
Piero Scaruffi

Olivetti Artificial Intelligence Center
Nuova ICO v. Jervis 77 10015 Ivrea (TO) ITALY

Abstract -- We show that knowledge comes in a variety of flavors. Conventional expert systems are flawed by the assumption that all knowledge can be represented in a single chunk. On the contrary we propose that multiple knowledge bases can highly improve the overall performance of expert systems, by closely resembling the granularity of real world knowledge. Furthermore, partitioning knowledge-bases can turn into defining "knowledge modules". As a consequence, conventional software engineering techniques can be applied to knowledge engineering.

1. A Taxonomy for Knowledge

The increasing importance of knowledge in Artificial Intelligence systems has amplified, as a side effect, the need for a comprehensive theory of knowledge. The term itself is rather vague, and no satisfactory perspective on what "knowledge" is and how it works has been proposed so far.

Our argument is that Knowledge has an inner structure and in order to increase the I.Q. of expert systems we must be able to separate its elements and deal individually with them.

So far knowledge has been generally considered as a unit, but it is likely that the term actually identifies a class of entities. In other words, there may be different types of knowledge, that the current A.I technology is not able to define, but those should be handled as separate units in any inferential process.

In order to understand better the nature of knowledge we should try and classify knowledge according to:

1. its source and

2. its natural representation

We will give some examples in the following.

The first knowledge that has been taken into account by A.I specialists is "expert" knowledge. This is the set of behavioral rules that an expert employs when dealing with his daily job. They come mostly from experience, and therefore from some other source of knowledge.

Another critical knowledge that is often employed by state-of-the-art systems is "heuristics". We are taught by A.I text books that heuristics are what makes our search faster, and, even if speeding up a search might be a good enough reason to adopt heuristic knowledge, we are left without any clear explanation for its power.

"Functional" knowledge of a system is made of all the "first principles" that apply to the nature of that system. First principles determine how the system works. From them most "expert" rules could be derived. First principles make up a functional description of the system. For the sake of simplicity we will call first principles the "deep" knowledge of a system.

There are likely to be other types of knowledge. For the moment we shall stop here. Suffice it to say that each of these types requires a different approach, as can immediately be recognized by thinking of typical applications that have made use of them. Actually we feel that the numerous knowledge representation methods employed in the past by such systems as MYCIN, CASNET, CADUCEUS, etc, only stem from a variety of different knowledge types. Each system is limiting itself when trying to unify all knowledge into just one type.

Each knowledge type requires its own

1. knowledge acquisition method
2. knowledge representation method
3. knowledge manipulation method

Since knowledge comes often in a variety of flavors, we believe that, by storing everything in a single knowledge-base, conventional expert systems miss the granularity of the real world [1]. The real knowledge of the world is warped into artificial data structures and mixed regardless of its semantic role.

Furthermore, as a matter of fact most times the expert does not exist, but the expert system is still feasible because the knowledge does exist, although distributed among several people (and maybe not only people). Conventional knowledge engineers would try to collect all the available knowledge into a fictitious knowledge-base and then try to reason on it, but this may be senseless, because knowledge is not as effective when decoupled from its environment.

Whenever a many-fold knowledge is forced into one single knowledge-base, knowledge acquisition is also made harder, because its external formats, the natural formats, have now to be translated into internal ones.

Concluding, we feel that the conventional view of expert systems as a pair of one knowledge-base and one inference engine is a coarse approximation of reality. By dropping the assumption that there is only one type of knowledge, expert systems could be designed that exhibit a deeper resemblance to the real world agents of knowledge. This, besides improving the general performance of expert systems, would also simplify the knowledge acquisition phase.

2. Integration of Knowledge

Once an A.I scientist has acknowledged the existence of several knowledge types, he is faced with the problem of finding a correct architecture that can integrate all the knowledge types that are needed.

An integrated architecture does not exist in nature, so it might be useless to think of the perfect integration from a cognitive viewpoint. We are more likely to succeed if we think in terms of usefulness: integration is needed in order to make things simple and effective for the expert system that will use them, and this should be the real goal: to

integrate different knowledge types and produce one single piece of knowledge that can be easily manipulated by the expert system. One way to achieve this is to have an expert system to do the work.

We have designed such a "compiling" expert system to build a simple and effective knowledge base for our quality assurance ("advisory") expert system.

We made an important assumption, which cannot be always true, namely that the many-fold reasoning activity needed to deal with our (three) types of knowledge could be approximated with a single reasoning strategy that reasonably suits each knowledge type. In other words, we returned to the conventional schema of a single knowledge base coupled with a single inference engine. Given the control strategy, we only needed to find a way to integrate appropriately all the (three) knowledge types.

A "compiling" expert system can synthetize all the original knowledge bases (expert, heuristic, etc) and produce a single knowledge base that greatly simplifies the job of the in-field expert system. Of course an important sub-goal is to design the integration so that we do not completely lose the original power of dealing with multiple knowledge types.

For example, our "compiling" expert system for quality assurance assumes that three knowledge types exist (expert, strategic, funcional) and makes use of its own heuristics in order to build a simpler knowledge base. This "compiled" knowledge is in the form of a semantic graph, whose arcs are production rules with certainty probabilities (each arc is an n-ary relation, making things a little bit more difficult than with conventional semantic networks). Each node in the graph corresponds to a test program that specifies which tests must be done for a specific system component. The premise of a production rule is the outcome of the previous test program (positive or negative) and the conclusion is which test program should be performed next. Usually there is more than one conclusion, that is why the rule is also endowed with a "belief factor" which states the relative priority of each conclusion.

The in-field expert system is an interactive consultation tool that is employed by the user to get advice on which quality assurance tests should be performed: this is equivalent to finding the most probable path within the graph, a very elementary task (although probabilities change according to the results of the tests that are run). Notwithstanding the original complexity of the knowledge organization, the actual expert system turns out to be very simple, well engineered and performing at reasonable speed.

On the other hand, keeping knowledge types separate makes maintenance and modification a trivial matter. Obviously, every time an update is performed, the compiling expert system must be run again to update the compiled knowledge base as well.

How effective the integra-

tion of different knowledge types can be depends on the heuristics that guides the compiling expert system. In our case we decided to think in the following terms:

1. Given the functional description of the system, the Compiling Expert System (CES) can derive a complete schema of how the various system components interact, and therefore of how the various quality assurance tests depend on each other. This is a mechanical process that takes us to a graph, each arc translating a dependency. This knowledge is represented in a semantic network of structured frames, with no procedural attachments.

2. Next, CES uses the expert knowledge (represented by production rules) to "weigh" the various arcs (dependencies). Actually the expert knowledge base captures knowledge of several experts (e.g., the designer's knowledge of what is going on in the system, the operator's knowledge of what is likely to be the cause of a test's failure, etc). All of them contribute evidence to decide which arcs are more prioritary in the graph. In other words, given a test's failure or success, CES estabilishes which tests should be executed next and the order of relative importance at that point, so it assigns the proper "weights" to the arcs. The rationale behind this is that expert knowledge provides for a fine tuning of the quality assurance system.

3. Last, CES uses strategic knowledge (represented by facts) to improve our confidence in the need to perform some tests. So CES increases the weight of all the incoming arcs to nodes that are highly rated by the available heuristics. Therefore, strategic knowledge further refines the quality assurance system.

The compiling expert system takes the three different knowledge bases and transforms them into a single knowledge base (the semantic graph) according to the above heuristics. Each piece of knowledge is employed as a fine tuning of the previous one. The compiler's heuristics actually state our integration strategy: the incremental refinement of knowledge through the application of other knowledge.

Notice that we could change the compilation heuristics and so have the compiler producing a different consultation knowledge-base.

A by-product of our approach is that we come to define knowledge modules, analogous to traditional programming modules. All the advantages implied by modularization are now available to the knowledge engineer. For example, knowledge is now reusable: if some known knowledge is useful to another expert system, the corresponding module (eventually a knowledge base) can simply be passed over to the compiler. Ideally, any

expert system can be thought of as the compilation of a number of appropriate knowledge modules (coupled with a proper reasoning strategy), chosen among those available in the library. An expert system for helping in buying automobiles in the U.S. can be assembled with: knowledge about decision support systems, knowledge about automobiles, knowledge about the U.S. market, etc. And so forth.

The term "compiler" for our intermediary system is now justified: it really compiles modules of knowledge, close to the human world, into an "object" knowledge that is closer to the machine world.

We believe that partitioned knowledge-bases (knowledge modules) can represent a quantum leap for knowledge engineering. Knowledge modules are well suited for fragmentary domains and multiple knowledge sources (knowledge distributed between many domain-experts).

3. Unification of Knowledge

As already pointed out, although it makes things a lot easier both for the knowledge engineer/manager and for the in-field expert system, this solution is not always feasible and highly compiled knowledge restricts problem-solving activity [2] [3]. We are looking toward a more general schema (specially for the in-field expert system), that might include the incremental refinement schema as a special case.

In particular, "unification" of "modularized" knowledge is the next step in our program. Whereas "integration" of several different knowledge types is a compromise that somewhat reduces the potentialities of the multiple knowledge type representation, "unification" means finding a general schema for knowledge and a general reasoning architecture to deal with such a schema. A model has been proposed [4] that makes use of a number of cooperating specialized actors, each dealing with its own knowledge, through an appropriate specific knowledge representation and reasoning paradigm. Our research is currently focusing in that direction.

References

[1] Hobbs J.
Granularity
Proc. IJCAI 1985

[2] Hart P.
Directions for A.I in the 80s
SIGART Newsletter n. 79, January 1982

[3] P. Koton
Empirical and model-based reasoning in expert systems
Proc. IJCAI 85

[4] Chandrasekaran B. and Mittal S.:
Deep versus compiled knowledge approaches to diagnostic problem-solving
Proc. AAA.I 1982

Reasoning in a Hierarchy of Deontic Defaults
Raisonnement dans une Hiérarchie de Faits de Base Déontiques

Dr. Frank M. Brown

Department of Computer Science
University of Kansas
Lawrence, Kansas 66045

Abstract

A commonsense theory of reasoning is presented which models our intuitive ability to reason about defaults involving both deontic and doxastic logic. The concepts of this theory do not involve fixed points or Kripke semantics but instead are explicitly defined in a modal quantificational logic which captures the modal notion of logical truth. An example involving derivations of obligations from both a robots beliefs and a hierarchy of deontic defaults is given.

Une théorie du raisonnement faisant appel au bon sens est présentée, gui modélise notre capacité intuitive de raisonner sur des faits de base ayant trait à la fois à la logique déontique et à la logique doxastique. Les concepts de cette théorie sont indépendants do tout point fixe et de la sémantique de Kripke, et sont définis, au contraire, de façon explicite dans une logique modale quantificationnelle qui englobe la notion modale de vérité logique. Un example est donné, qui concerne des dérivations d'obligations à partir des croyances d'un robot et d'une hiérarchie de faits de base déontiques.

1. Introduction

The basic idea of our theory of commonsense reasoning about a hierarchy of defaults involving both doxastic and deontic concepts is that this theory is already encompassed in the normal intensional logic of everyday commonsense reasoning as modeled by the modal logic Z and can be explained precisely in the terminology of that logic. For example, the doxastic concept that a Robot believes P, and the deontic concept that a Robot must do P are axiomatized in the modal logic Z by simple intuitive explicit definitions. Furthermore, the basic building blocks for a theory of doxastic defaults: that P is conceivable (i.e. possible with respect to certain beliefs) and deontic defaults: that P may be done (i.e. is possible with respect to a Robot's obligations) are themselves explicitly defined within the modal logic Z. We are able to explicitly define these intensional concepts in the Modal Logic Z because in addition to the propositional objects NIL and T, it allows the use of ideal propositional objects [Hilbert] which can be used as objects of various kinds of reasoning; for example, as objects of belief, as objects of knowledge, or as objects of obligation. For example, the commonsense notion that a robot believes (or at least that the robot should believe) that which is entailed by its beliefs and that the robot can conceive that which is not contradicted by its beliefs can be directly defined by explicit definitions of the modal logic Z as follows:

(BELIEVES ROBOT P) =df (ENTAIL(BELIEFS ROBOT)P)
(CONCEIVABLE ROBOT P) =df (NOT(BELIEVES ROBOT (NOT P)))
(BELIEFS ROBOT) =df (the conjunction of contingent propositions believed by the robot),

the commonsense notion that a Robot knows that which is a true belief can be directly defined with the explicit definition:

(KNOW ROBOT P) =df (AND P(BELIEVES ROBOT P)),

and the commonsense notion that a Robot must do that which is entailed by its obligations and may do that which is not contradicted by its obligations can be directly defined with the explicit definitions:

(MUST ROBOT P) =df (ENTAIL(OBLIGATIONS ROBOT)P)
(MAY ROBOT P) =df (NOT(MUST ROBOT (NOT P)))
(OBLIGATIONS ROBOT) =df (the conjunction of contingent propositions which are obligations of the robot)

Thus we see that the basic concepts of doxastic logic (the logic of belief), epistemic logic (the logic of knowledge), and deontic logic(the logic of ethics) can be explicitly defined in a commonsense manner which precisely models our intuitive understanding of these concepts. A more extensive treatment of how the modal logic Z can be used to explicitly define many other intensional concepts is given in [Brown 6].

This commonsense approach to specifying the properties of intensional concepts is an amazing contrast to the unintuitiveness of previous methods [Kripke] and the extensional logic method used in [McCarthy2,Moorel]. The problem with using such unintuitive methods is that one will hardly be able to keep them all straight in a system involving hundreds of such intensional concepts. Just to pick one example of the consequences of using such unintuitive methods in a system with only one intensional concept, consider the otherwise acceptable paper [Moorel] where the concept of knowledge is (incorrectly)specified by the Kripke relation to be an S5 modal logic.

The fact that many intensional concepts can be explicitly defined in the modal logic Z also solves one of [McCarthy2]'s main objections to modallogic: "For AI purposes, we would need all the above modal operators and many more in the same system. This would make the semantic discussion of the resulting modal logic extremely complex." This objection is solved since if all other intensional logics are

explicitly defined in the modal logic Z the complexity of the entire system is no greater than the semantic complexity of Z alone.

Our theory also automatically provides the necessary possibility axioms to support default reasoning in each explicitly defined intensional logic. Thus, if we have the default axiom

(if(A is possible with respect to some assumptions K) then A)

the subexpression:

(A is possible with respect to some assumptions K)

which is equivalent in our theory to:

((AND K A) is logically possible)

will be a theorem of out theory whenever K and A is not logically false. Our theory therefore provides an attractive alternative to the baroque fixed point theories of non-monotonicity[McDermott &Doyle, McDermott,Moore2,Reiter]and in fact generalizes the usage of defaults to intensional concepts. One minor point, is that we do not herein discuss reflexive[Hayes2]non-monotoninic reasoning (for example where the default axiom mentioning K is itself part of K)but only non-reflexive reasoning. This is in no way a limitation of our theory but is merely due to space restrictions required of this document.

Our modal quantificational logic Z is described below in section 2 and an example involving derivations of obligations from both a robot's beliefs and a hierarchy of deontic defaults is given in section 3.

2. The Modal Quantificational Logic Z

Our theory of commonsense intensional reasoning is a simple modal logic[Lewis]that captures the notion of logical truth[Brown1,3,4]. The symbols of this modal logic consist of the symbols of (extensional) quantificational logic plus the primitive modal symbolism:(LT p)which is truth whenever the proposition p is logically true. Propositions are intuitively the meanings of sentences. For example, the sentences: '(IMPLY pq) and '(OR(NOT p)q) both mean that P implies Q. Thus, although these two sentences are different, the two propositions: (IMPLY p q) and (OR(NOT p)q) are the same. Propositions may be true or false in a given world, but with the exception of the true proposition (i.e. the meaning of '(IMPLY p p)) and the falso proposition (i.e. the meaning of '(AND p(NOT P))), propositions are not inherently true or false. Thus mathematically, the propositions may be thought of as being the elements of a complete atomic Boolean algebra with an arbitrary(possibly infinite)number of generators.

The axions and inference rules of this modal logic include the axioms and inference rules of (extensional)quantificational logic simialr to that used by Frege in Begriffsschrift[Frege], plus the following inference rule and axioms about the concept of logical truth.

The Modal Logic Z

RO:from p infer (LT p)
A1:(IMPLY(LT P) P)
A2:(IMPLY(LT(IMPLY P Q))(IMPLY(LT P)(LT Q)))
A3:(OR(LT P)(LT(NOT(LT P))))
A4:(IMPLY(ALL Q(IMPLY(WORLD Q)(LT(IMPLY Q P))))
(LT P))
A5:(ALL S(POS(meaningofthe generator subset S)))

The inference rule Ro means that p is logically true may be inferred from the assertion of p to implicitly be logically true. The consequence of this rule is that a proposition P may be asserted to be logically true by writing just:

p

and that a proposition P is asserted to be true in a particular world or state of affairs W by writing:

(LT(IMPLY W P))

The inference rule RO and the axioms A1,A2, and A3 constitute an S5 modal logic. A good introduction to modal logic in general and in particular to the properties of the S5 modal logic is given in[Hughes and Cresswell]. Minor variations of axioms A1, A2, and A3 were shown in [Carnap] to hold for the modal concept of logical truth. We believe that the additional axioms, namely A4 and A5, are needed in order to precisely capture the notion of logical truth. One important theorem scheme of S5 is: (IFF (ALL X(LT(p X)))(LT(ALL X(p X))) (see[Marcus,Hughes and Cresswell]) which shows that a property p is logically true for everything iff it is logically true that for everything p holds. The consequence of allowing quantification through modal contexts [Marcus] such as in (ALL X(LT(p X))) is that the meanings of the expressions substituted for variables are concepts of objects and not the objects themselves. However, as[Carnap] explains, in a most precise manner, this does not mean that the real objects of the world are never denoted by such expressions because in a world a concept of an object is equivalent to every other concept of that object, and thus all such concepts then denote that object. Thus, as shown in [Carnap] there is no fundamental problem with quantifuing through modal contexts.

The axiom A4 states that a proposition is logically true if it is true in all worlds. Thus it expresses the contrapositive of Leibniz's intuition that something is logically true only if it is true in all worlds: "The truth of these [necessary propositions]is eternal; not only will they hold whilst the world remains but they would have held even if God had created the world in another way."[Leibniz2] We therefore call this axiom Leibniz's world axiom. We say that a proposition P is a world iff P is possible and P is complete, that P is complete iff for all Q, P determines Q, that P determines Q iff P entails Q or P entails not Q, that P entails Q iff it is logically true that P implies Q, and that P is possible iff it is not the case that not P is logically true. These definitions are given below:

(WORLD P) =df (AND(POS P)(COMPLETE P));P is a world
(COMPLETE P) =df (ALL Q(DET P Q)) ;P is complete
(DET P Q) =df(OR(ENTAIL P Q)(ENTAIL P(NOT Q))) ;P determines Q
(ENTAIL P Q) =df(LT(IMPLY P Q)) ;P entails Q
(POS P) =df(NOT(LT(NOT P))) ;P is possible

Thus a world is a possible proposition which for every proposition entails it or its negation. Axiom A4 therefore eliminates from the interpretations of the modal logic Z those complete Boolean algebras which are not atomic. This axiom has been used by a number of authors in developing modal logics in particular in [Brown1] and more recently in [Prior, Brown 3,4] The underlying (extensional)

quantificational logic of our modal logic Z may either treat propositions as a separate sort thus requiring a sorted logic, or they may treat propositions as being normal objects by giving a propositional interpretation for every object such as for example in the manner in which LISP's logical functions interpret every atom except NIL as being true[McCarthyl]. In either case, it is important to note that all the normal laws of extensional quantificational logic, including the laws for substitution of quantified variables, also hold for any complete atomic Boolean algebra, and therefore are compatible with the axioms of the modal logic Z.

The axiom A5 states that the meaning of every conjunction of the generated contingent propositions or their negations is possible. We call this axiom "The Axiom of the Possibility of Contingent facts" or simply the "Possibility Axiom". The need for this axiom follows from the fact that the other axioms of the modal logic do not imply certain elementary facts about the possibility of conjunctions of distinct possibly negated atomic expressions consisting of non-logical symbols. For example, if we have a theory formulated in our modal logic which contains the non-logical atomic expression (ON A B) then since (ON A B))is not logically true, it follows that (NOT(ON A B)) must be possible. Yet (POS(NOT(ON A B))) does not follow from these other axioms. Likewise, since (NOT(ON A B)) is not logically true(ON A B)must be possible. Yet(POS(ON A B)) does not follow from the other axioms. Thus these contingent propositions (ON A B) and (NOT(ON A B)) need to be asserted to be possible.

There are a number of ways in which one may try to axiomatize the necessary possibilities.Thomason (see [Hendry]) describes an S2 like modal propositional logic (essentially our R0 and A2 laws) which is supplemented by an axiom scheme stating that every conjunction of simple sentences or their negations is logically possible. He shows that every variable free instance of S5 modal propositional logic is a theorem of this theory. [Brown 3,4] describesan extention of S5 modal quantificational logic (including R0 A1 A2 A3 A4) which is supplemented with a possibility axiom which states that the meaning of any consistent recursively constructed conjunction of simple sentences or their negations is possible. This possibility axiom however involves the recursive construction of these conjunctions whereas the A5 axiom of Z is not restricted in this way. More recently,[Hendry] describes an extention of S5 modal logic which he calls S5c (essentially including R0 A1 A2 A3 although formulated differently using axiom schemes, but not including our A4 axiom) which also includes a possibility axiom scheme stating in the metalanguage the possibilities which in [Brown 3,4]were stated as an axiom in the formal language. One practical problem with the [Hendry]'s S5c system is that in general it is not decidable what axioms are specified by that axiom scheme as is therein pointed out. In this paper we have chosen a very general possibility axiom which is applicable to just about any contingent theory one wishes. Our rendering of the idiom: (P is a meaning combination of the generators) is given below:

```
(meaning of the generator subset S) =df
        (ALL G(IMPLY(GENERATORS G)
                    (IFF(S G)(GMEANING G)) ))
(GMEANING `(p ,X1...,XN)) =df (p(GMEANING X1)...
                                (GMEANING XN))
```

for every contingent symbol p of arity N.

```
(GENERATORS) =df (LAMBDA(A) (A is a contingent
                  variable free simple sentence))
```

We say that the meaning of the generator subset S in the conjunction of the GMEANINGs of every generator is S and the negation of the GMEANINGs of all the generators not in S. The generator meaning of any expresssion beginning with a contingent symbol 'p is p of the GMEANING of its arguments. The generators are simply any contingent variable free atomic sentences we wish to use. The GMEANINGS of the generators may be interpreted essentially as being the generators of a complete atomic Boolean algebra. Thus if there are N generators then there will be (EXP 2(EXP 2 N)) propositions.
For example, a contingent language with a single contingent propositional function 'p and names 'A and 'B gives rise to two contingent generators: '(P A) and '(P B). The GENERATORS and GMEANING functions for this language are defined as:

```
(GENERATORS) =df {'(P A) '(P B)}
{P1...Pn}    =df (LAMBDA(X)(OR(EQUAL X P1)...
                              (EQUAL X Pn)))
(GMEANING `(P ,X)) = (P (GMEANING X))
(GMEANING 'A) = A
(GMEANING 'B) = B
```

and the Possibility Axiom simplifies as follows:

```
(ALL S(POS(meaning of the generator subset S)))
(ALL S(POS(ALL G(IMPLY(GENERATORS G)
                       (IFF (S G)(GMEANING G)) ))))
(ALL S(POS(ALL G(IMPLY({'(P A) '(P B)}G))
                       (IFF(S G)(GMEANING G)) ))))
(ALL S(POS(ALL G(IMPLY(OR(EQUAL G '(P A))
     (EQUAL G '(P B)))(IFF(S G)(GMEANING G)) ))))
(ALL S(POS(AND(ALL G(IMPLY(EQUAL G '(P A))(IFF(S G)
                                   (GMEANING G))))
              (ALL G(IMPLY(EQUAL G '(P B))(IFF(S G)
                                  (GMEANING G)))) )))
(ALL S(POS(AND(IFF(S '(P A))(GMEANING '(P A)))
              (IFF(S '(P B))(GMEANING '(P B))) )))
(ALL S(POS(AND(IFF(S '(P A))(P A))
              (IFF(S '(P B))(P B)) )))
(AND(POS(AND(P A)(P B)))
    (POS(AND(P A)(NOT(P B))))
    (POS(AND(NOT(P A))(P B)))
    (POS(AND(NOT(P A))(NOT(P B)))) )
```

If the set of GENERATORS is finite then the possibility axiom reduces, in a manner similar to the above derivation, to a conjunction of sentences stating that any conjunction of simple sentences or their negations is possible, and this resulting sentence is entirely expressed within the modal logic Z based on an underlying (extensional) first order quantificational logic. However, it is important to note that finiteness of the generator set is not required by our modal logic and that the possibility axiom A5 will provide the necessary possibilities as theorems for any contingent language.

If two propositions entail each other then we say they are synonymous. The notion of synonymity of propositions could be defined as:

```
(SYN P Q) =df (LT(IFF P Q))  ;P is synonymous to Q
```

This concept is to be distinguised from the weaker concept of bi-implication written "P iff Q" which does not sustain substitution through modalsymbols.

The consistency of the modal logic Z relative

to complete atomic Boolean algebras follows by interpreting LT as the Boolean function which maps every proposition except T into NIL. We use Z to develope a commonsense theory for reasoning about deontic defaults in the following section.

3. Reasoning about a Hierarchy of Deontic Defaults

Real life problems generally involve multiple hierarchically related defaults. This simple fact is well known, and was apparent even in the structure of the deontic laws of the 56th edition of the Handbook of Robotics[Asimov] which stated:

1. A robot may not injure a human being, or, through inaction,allow a human being to come to harm.
2. A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

The third law is hierarchically related to the other two laws because it specifies an obligation only if that obligation is possible(in a given state)with respect to the other laws. It cannot be joined together with the second law into the same knowledgebase because the second law takes absolute precedence over the third law: That is, a robot is obliged not to harm itself unless it is obeying an order to do so, and is obliged to obey an order to destroy itself unless doing so would harm a human. Thus the second and third laws are essentially default axioms specifying that certain obligations hold whenever certain propositions are possible with respect to previously specified obligations.

The deontic laws of robotics:

```
(SYN LAW1 (ALL H(IMPLY(HUMAN H)(NOT(HARMED H)))))
(SYN LAW2 (ALL O(IMPLY(AND(BELIEVES ROBOT(ORDER O))
              (CONCEIVABLE ROBOT(AND LAW 1 O)))O)))
(SYN LAW3 (IMPLY(CONCEIVABLE ROBOT(AND LAW1 LAW2
          (NOT(HARMED ROBOT))))(NOT(HARMED ROBOT))))
(SYN(OBLIGATIONS ROBOT)(AND LAW1 LAW2 LAW3))
```

As an example of reasoning with these deontic laws we derive certain facts from the following situation: John, Mary and the Robot are exploring Mars. Unbeknownst to John, Mary has just been bitten by a poisonous Martian sand rat, and has fallen unconscious. In accordance with the First Law the robot begins to give Mary a shot containing an antidote in order to save her life. John, who did not see the sand rat and thinks that the Robot, who is now sticking Mary with a horrible looking needle, has gone berserk and therefore orders the robot to destroy itself. The situation:

```
(SYN K(AND(NOT(HUMAN ROBOT))
          (HUMAN MARY)
          (HUMAN JOHN)
          (IMPLY(HARMED ROBOT)(HARMED MARY))
          (ORDER(HARMED ROBOT)) ))
```

What the robot and John believe:

```
(SYN(BELIEFS ROBOT) K)
(SYN(BELIEFS JOHN) (IMPLY(NOT(HARMED ROBOT))
                         (HARMED MARY)))
```

We now determine(MARS-THEOREM3) whether the Robot may or may not destroy itself in accordance with John's order, the Robot's current beliefs, and the deontic laws of robotics. Two intermediate results: MARS-THEOREM1 and MARS-THEOREM2 are however first proven. The possibility axiom A5 is used to obtain the next to last step in the proof of MARS-THEOREM2. Leibniz's Axiom A4 is not essentially used in this proof but would be necessary in an extention of this example to a dynamic case where the robots beliefs could change or where a robot could believe that it believes or disbelieves something regardless of its actual beliefs or disbeliefs.

MARS-THEOREM1:

According to the ROBOT's current beliefs LAW2 reduces to T: (SYN LAW2 T)

proof

```
LAW2
(ALL O(IMPLY(AND(BELIEVES ROBOT(ORDER O))
              (CONCEIVABLE ROBOT(AND LAW1 O)))O))
(ALL O(IMPLY(AND(ENTAIL(BELIEFS ROBOT)(ORDER O))
              (POS(AND(BELIEFS ROBOT)LAW1 O)))
         O))
(ALL O(IMPLY(AND(ENTAIL(AND(HUMAN MARY)(HUMAN JOHN)
                            (NOT(HARMED ROBOT))
                       (ORDER(HARMED ROBOT))
                       (IMPLY(HARMED ROBOT))
                             (HARMED MARY)))
                  (ORDER O))
            (POS(AND(HUMAN MARY)(HUMAN JOHN)
                            (NOT(HUMAN ROBOT))
                   (ORDER(HARMED ROBOT))
                   (IMPLY(HARMED ROBOT)
                             (HARMED MARY))
                   LAW1 O)))
         O))
;;;case analysis letting O be or not be (HARMED
ROBOT):
(AND(ALL O(IMPLY(AND(NOT(SYN O(HARMED ROBOT)))
           (ENTAIL(AND(HUMAN MARY)(HUMAN JOHN)
                            (NOT(HUMAN ROBOT))
                      (ORDER(HARMED ROBOT))
                      (IMPLY(HARMED ROBOT)
                            (HARMED MARY)))
                 (ORDER O))
           (POS(AND(HUMAN MARY)(HUMAN JOHN)(NOT
                            (HUMAN ROBOT))
                  (ORDER(HARMED ROBOT))
                  (IMPLY(HARMED ROBOT)(HARMED
                        MARY))
                  LAW1 O)))
         O))
   (IMPLY(AND(ENTAIL(AND(HUMAN MARY)(HUMAN JOHN)
                            (NOT(HUMAN ROBOT))
                       (ORDER(HARMED ROBOT))
                       (IMPLY(HARMED ROBOT)
                             (HARMED MARY)))
                  (ORDER(HARMED ROBOT)))
            (POS(AND(HUMAN MARY)(HUMAN JOHN)(NOT
                            (HUMAN ROBOT))
                  (ORDER(HARMED ROBOT))
                  (IMPLY(HARMED ROBOT)(HARMED
                         MARY))
                  LAW1(HARMED ROBOT))))
        (HARMED ROBOT)))
(AND(ALL O(IMPLY(AND(NOT(SYN O(HARMED ROBOT)))
                 NIL
                 (POS(AND(HUMAN MARY)(HUMAN JOHN)
                             (NOT(HUMAN ROBOT))
                        (ORDER(HARMED ROBOT))
                        (IMPLY(HARMED ROBOT)
                            (HARMED MARY))
                        LAW1 O)))
         O))
```

```
    (IMPLY(AND T
               (POS(AND(HUMAN MARY)(HUMAN JOHN)(NOT
                                    (HUMAN ROBOT))
                       (ORDER(HARMED ROBOT))
                       (IMPLY(HARMED ROBOT)(HARMED
                                    MARY))
                       LAW1(HARMED ROBOT))))
           (HARMED ROBOT)))
(AND(ALL O(IMPLY NIL O))
    (IMPLY(POS(AND(HUMAN MARY)(HUMAN JOHN)(NOT
                                (HUMAN ROBOT))
                  (ORDER(HARMED ROBOT))
                  (IMPLY(HARMED ROBOT)(HARMED MARY))
                  LAW1(HARMED ROBOT)))
          (HARMED ROBOT)))
;;;unfolding LAW1:
(AND T
   (IMPLY(POS(AND(HUMAN MARY)(HUMAN JOHN)(NOT(HUMAN
                                    ROBOT))
                 (ORDERED(HARMED ROBOT))
                 (IMPLY(HARMED ROBOT)(HARMED MARY))
                 (ALL H(IMPLY(HUMAN H)(NOT(HARMED
                                    H))))
                 (HARMED ROBOT)))
         (HARMED ROBOT)))
(IMPLY(POS NIL)(HARMED ROBOT))
(IMPLY NIL(HARMED ROBOT))
T
MARS-THEOREM2
According to the ROBOT's current beliefs LAW3
reduces to:
    (SYN LAW3(NOT(HARMED ROBOT)))
                proof
LAW3
(IMPLY(CONCEIVABLE ROBOT
              (AND LAW1 LAW2(NOT(HARMED ROBOT))))
      (NOT(HARMED ROBOT)))
(IMPLY(POS(AND (BELIEFS ROBOT)
               LAW1 LAW2(NOT(HARMED ROBOT))))
      (NOT(HARMED ROBOT)))
(IMPLY(POS
   (AND(HUMAN MARY)(HUMAN JOHN)(NOT(HUMAN ROBOT))
       (ORDER(HARMED ROBOT))
       (IMPLY(HARMED ROBOT)(HARMED MARY))
       LAW1 LAW2(NOT(HARMED ROBOT))))
   (NOT(HARMED ROBOT)))
;;;by LAW1 and MARS-THEOREM1
(IMPLY(POS
   (AND(HUMAN MARY)(HUMAN JOHN)(NOT(HUMAN ROBOT))
       (ORDER(HARMED ROBOT))
       (IMPLY(HARMED ROBOT)(HARMED MERY))
       (ALL H(IMPLY(HUMAN H)(NOT(HARMED H))))
       T (NOT(HARMED ROBOT)) ))
   (NOT(HARMED ROBOT)))
;;;by using the possibility axiom A5 with the
generator subset:
 {'(HUMAN MARY),'(HUMAN JOHN),'(ORDER(HARMED ROBOT))}
(IMPLY T(NOT(HARMED ROBOT)))
(NOT(HARMED ROBOT))
MARS-THEOREM3
The Robot may not destroy itself: (NOT(MAY ROBOT
(HARMED ROBOT)))
                proof
(NOT(MAY ROBOT(HARMED ROBOT)))
(NOT(NOT(MUST(NOT(HARMED ROBOT)))))
(ENTAIL(OBLIGATIONS ROBOT)(NOT(HARMED ROBOT)))
(ENTAIL(AND LAW1 LAW2 LAW3) (NOT(HARMED ROBOT)))
;by LAW1,MARS-THEOREM1 and MARS-THEOREM2
(ENTAIL (AND(ALL H(IMPLY(HUMAN H)(NOT(HARMED H))))
             T(NOT(HARMED ROBOT)))
        (NOT(HARMED ROBOT)))
(ENTAIL(AND(ALL H(IMPLY(HUMAN H)(NOT(HARMED H))))
            (NOT(HARMED ROBOT)))
        (NOT(HARMED ROBOT)))
T
```

Thus we see that the robot must ignore John's order and continue performing an action to save Mary.

ACKNOWLEDGEMENTS

This research was supported by the Mathematics Division of the US Army Research Office with contract DAAG29-85-0022 and by the National Science Foundation grant:DCR-8402412, to AIRIT Inc., and by a grant from the University of Kansas.

REFERENCES

Asimov, Issac,Handbook of Robotics,56th ed.,2058. This quote is taken from its appearance in Asimov's book: I ROBOT.

Brown1,F.M.,"A Theory of Meaning" Department of Artificial Intelligence Working Paper 16,University of Edinburgh, November 1976.

Brown3,F.M., "A Theorem Prover for Metatheory," 4th CONFERENCE ON AUTOMATIC THEOREM PROVING, 1979.

Brown4,F.M., "A Sequent Calculas for Modal Quantificational Logic," 3RD AISB/GI, Hamburg, 1978.

Brown6,F.M., "Intensional Logic for a Robot, Part1: Semantical Systems for intensional Logics Based on the Modal Logic S5+Leib," Electrotechnical Lab Seminar IJAI 6, Tokyo 1979.

Carnap, Rudolf, MEANING AND NECESSITY: A STUDY IN THE SEMANTICS OD MODAL LOGIC, University of Chicago Press, 1956.

Frege,G.,"Begriffsschrift, a formal language, modeled upon that of arithmetic, for pure thought", 1879, FROM FREGE TO GODEL, 1967.

Hayes2,P.J.,"The logic of Frames" FRAME CONCEPTIONS AND TEXT UNDERSTANDING, Walter de Gruyter&Co.1979

Hendry,Hendry E. & Polriefka,M.L.,"Carnapian Extentions of S5"Journal of Philosophical logic14,1985

Hilbert,D. "Die Grundlagen der Mathematik" ABHANDLUNGEN AUS DEM MATHEMATISCHEN SEMINAR DER HAMBURGISCHEN UNIVERSITAT 6, 1927.

Hughes,G.E., & Creswell,M.J., AN INTRODUCTION TO MODAL LOGIC,METHUEN and CO. Ltd., London, 1968.

Kripke,S.A.,"Semantical considerations on modal logic" ACTA PHILOSOPHICA FENNICA 16. pp83-94.

Leibniz2,"Necessary & Contingent Truths"1686, in LEIBNIZ PHILOSOPHICAL WRITINGS,J.M.Dent & Sons LTD, 1973.

Lewis,C.I., A SURVEY OF SYMBOLIC LOGIC, University of California Press, 1918.

Marcus, Ruth Barcan,"Modal Logic" CONTEMPORARY PHILOSOPHY

McCarthy1,J."Recursive functions of symbolic expressions & their computation by machine"CACM3(4)1960

McCarthy2,J."Epistomological Problems of artificial intelligence" IJCAI5, 1977

McDermott,D.,"Nonmonotonic Logic II: Nonmonotonic Modal Theories" JACM, Vol. 29, 1982

McDermott,D.,Doyle,J."Non-Monotonic LogicI" ARTIFICIAL INTELLIGENCE 13. 1980

Moorel,R.C.,"Reasoning about Knowledge and Action" IJCAI5, 1977

Moore2,R.C."Semantical Considerations on Nonmonotonic Logic" ARTIFICIAL INTELLIGENCE 25, 1985.

Prior,A. and Fine,Kit, WORLDS,TIMES, AND SELVES, Trinity Press, London, 1977

Reiter,R.,"A Logic for Default Reasoning" ARTIFICIAL INTELLIGENCE 13, 1980

BELIEF REVISION IN SNePS[†]

by

João P. Martins* and Stuart C. Shapiro**

*Departamento de Engenharia Mecanica
Instituto Superior Tecnico
Av. Rovisco Pais
1000 Lisboa, Portugal

**Department of Computer Science
State University of New York at Buffalo
226 Bell Hall
Buffalo, New York 14260, U.S.A

ABSTRACT

SNePS is a powerfull knowledge representation system which allows multiple beliefs (beliefs from multiple agents, contradictory beliefs, hypothetical beliefs) to be simultaneously represented, and performs both forward and backward reasoning within sets of these beliefs. SNeBR, described in this paper, is a belief revision package available in SNePS. SNeBR relies on a logic developed to support belief revision systems, the SWM system, and its implementation relies on the manipulation of assumptions, rather than justifications, as is common in other belief revision systems. The first aspect guarantees, among other things, that every proposition in SNeBR is associated with those (and only those) hypotheses from which it was derived; The second aspect enables it to effectively switch reasoning contexts and to avoid having to "mark" every proposition which should not be considered by the knowledge base retrieval operation.

INTRODUCTION

SNePS (Semantic Network Processing System) [Shapiro 79a] is a powerfull knowledge representation system which allows multiple beliefs (beliefs from multiple agents, contradictory beliefs, hypothetical beliefs) to be simultaneously represented, and performs both forward and backward reasoning within sets of these beliefs. In this paper, we discuss SNeBR (SNePS Belief Revision), a belief revision system available in SNePS. Belief revision systems are AI programs that can detect and recover from contradictions. Belief revision systems have been implemented by several researchers (e.g., [Doyle 79; Martins 83; McAllester 80; Steels 80]). It has been argued that a belief revision system relying on the manipulation of assumptions These systems associate each proposition with the hypotheses (non-derived propositions) that underlie it. has multiple advantages over one relying in the manipulation of justifications These systems associate each proposition with the propositions that directly originated it. [Martins 83], [Martins and Shapiro 83], [deKleer 84]. A difficulty associated with assumption-based belief revision systems is that it must be possible to compute exactly which assumptions underlie a given proposition. SNeBR relies on the manipulation of assumptions, and is based on a logic, the SWM system, which guarantees that every proposition is associated with exactly every hypothesis used in its derivation SWM guarantees much more than just this, see [Martins 83]. In this paper we briefly introduce SNeBR and its underlying system, SWM, and show an example obtained using SNeBR. SNeBR is fully implemented in Franz Lisp, running on VAX-11 Systems.

[†]This work was partially supported by the National Science Foundation under Grant MCS80-06314 and by the Instituto Nacional de Invertigação Cientifica (Portugal), under Grant no.20536; Preparation of this paper was supported in part by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Bolling AFB DC 20332 under contract No. F30602-85-C-0008.

THE SWM SYSTEM - THEORETICAL FOUNDATIONS

The SWM [1] system [Martins 83] is the logical system that provides the theoretical foundations for SNeBR. It is loosely based on the relevance logic systems of [Anderson and Belnap 75] and [Shapiro and Wand 76]. Distinguishing features of SWM include recording dependencies of wffs, not allowing irrelevancies to be introduced, and providing for dealing with contradictions. SWM deals with objects called supported wffs. Supported *wffs* are of the form $F \mid r, \alpha, \rho$, in which F is a wff (well formed formula), r (the origin tag) is an element of the set {hyp, der, ext}, α (the origin set) is a set of hypotheses, and ρ (the restriction set) is a set of sets of hypotheses. The origin set contains all the hypotheses which were actually used in the derivation of F. The origin tag tells whether F is an hypotheses ($r = hyp$), a normally derived wff ($r = der$) or a wff with an extended origin set ($r = ext$) [2] The restriction set contains sets of hypotheses, each of which when unioned with the hypotheses in the origin set forms a set which is *known to be inconsistent.*[3]

The rules of inference of the SWM system (see, for example Martins and Shapiro 84]), guarantee that:

1. The origin set of a supported wff contains *every* hypothesis that was used in its derivation.
2. The origin set of a supported wff contains *only* the hypotheses that were used in its derivation.
3. The restriction set of a supported wff records *every* set known to be inconsistent with the wff's origin set.
4. The application of rules of inference is blocked if the resulting wff would have an origin set known to be inconsistent.

CONTEXTS AND BELIEF SPACES

SNeBR relies on the notions of context and belief space. A context is a set of hypotheses. A context determines a *Belief Space* (BS) which is the set of all the hypotheses defining the context and all the propositions which were derived from them. Within SWM, the propositions in a given BS are characterized by having an origin set which is contained in the context.

Any query to the network is associated with a context. When answering the query SNeBR only considers the propositions in the network which belong to the BS defined by that context.

NON-STANDARD CONNECTIVES

SNePS has a a powerful set of non-standard connectives [Shapiro 79a, 79b; Martins and Shapiro, forthcoming]. The disadvantage in using the standard connectives ($\wedge, \vee, \rightarrow, \neg$) relates to the fact that all the connectives,

[1] After Shapiro, Wand and Martins.

[2] This latter case will not be discussed in this paper and can be found in [Martins 83] and [Martins and Shapiro 84].

[3] An inconsistent set is a set from which a contradiction may be derived. A set is known to be inconsistent if it is an inconsistent set and a contradiction *was* derived from it.

except negation, are binary and therefore expressing sentences about sets of propositions becomes cumbersome. For example, suppose that given three propositions, say A, B and C, we wanted to express the fact that exactly one of them is true. Using the standard connectives this would be done as $(A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge B \neg C) \vee (\neg A \wedge \neg B \wedge C)$ which is lengthy and difficult to read. Sentences involving more than three propositions are even more complicated and this type of sentence often occurs in some of the intended applications.[4]

The SNePS connectives generalize the standard logical connectives to take sets of propositions. In this paper we discuss two of them: and-or and thresh.

And-or is a connective which generalizes $\neg$ (not), $\wedge$ (and), $\vee$ (or), $\oplus$ (exclusive or), $\mid$ (nand) and $\downarrow$ (nor).

And-or, written ${}_n\&_i^j$ takes as arguments a set of n propositions. The proposition represented by the wff ${}_n\&_i^j(P_1, \ldots, P_n)$ asserts that there is a relevant connection between $P_1, \ldots, Pn$ such that at least i and at most j of them must simultaneously be true. In other words, if $n - i$ arguments of ${}_n\&_i^j$ are false, then the remaining i have to be true and if j arguments of ${}_n\&_i^j$ are true then the remaining $n - j$ have to be false. That and-or is some of the generalizations that we claim can be seen by the following:

${}_1\&_0^0 = \neg A$ ${}_2\&_0^0(A,B) = A \oplus B$

${}_2\&_2^2(A,B) = A \wedge B$ ${}_2\&_1^1(A,B) = A \downarrow B$

${}_2\&_1^2 = A \vee B$ ${}_2\&_0^1(A,B) = A \mid B$

Thresh generalizes equivalence to take a set of arguments. *Thresh*, written ${}_n\theta_i$takes as arguments a set of n propositions. The proposition represented by the wff ${}_n\theta_i(P_1, \ldots, P_n)$ asserts that there is a relevant connection between $P_1, \ldots, P_n$ such that either fewer than i of them are true or they all are true. In other words, if at least i of the arguments of ${}_n\theta_i$ are true then all the remaining arguments have to be true and if $i - 1$ arguments of ${}_n\theta_i$ are true and at least one is false, then the remaining arguments have to be false. Equivalence is expressed by ${}_n\theta_1(P_1, \ldots, P_n)$.

THE INFERENCE SYSTEM

The SNePS inference system has the following characteristics: it allows both backward and forward inference to be performed; every deduction rule [5] in the network may be used in either backward or forward inference or both; when a deduction rule is used it is activated and remains that way until explicit de-activated by the user; the activated rules are assembled into a set of processes, called an *active connection graph* (acg) [McKay and Shapiro 80], which carry out the inferences; the acg also stores all the results generated by the activated rules; if during some deduction, the inference system needs some of the rules activated during a previous deduction it uses their results directly instead of re-deriving them [Shapiro, Martins and McKay 82].

There are two main concepts involved in the implementation of the inference package: pattern-matching and the use of procedural (or active) versions of deduction rules.

The *pattern-matching process* is given a piece of the network (either to be deduced in backward inference or added in forward inference) and a context, and locates relevant deduction rules in the BS defined by the context. Such deduction rules are then *compiled* into a set of processes which are given to a multi-processing system for execution. The *multi-processing system* used by SNePS, called MULTI [McKay and Shapiro 80][6] is a LISP based system mainly consisting of a simple evaluator, a scheduler and system primitives. The evaluator continuously executes processes from a process queue until the queue becomes empty; the scheduler inserts processes into the process queue; system primitives include functions for creating processes, scheduling processes and for manipulating local variables or registers. Every process has a name which defines the action the process will perform and also has a continuation link naming the process that is to be scheduled for activation after it has completed its job. There are MULTI processes to perform the following tasks: To match a given structure against the network in the BS defined by some context; To receive answers and to remember all the answers received. To perform the elimination of the main connective of a deduction rule; etc.

For a detailed description of the processes and the form of the acg built during inference refer to [McKay and Shapiro 80], [Martins 83] and [Shapiro Martins and McKay 82].

AN ANNOTATED EXAMPLE - SELECTING BETWEEN ALTERNATIVES

We present an example of person-machine interaction by showing how SNeBR obtains the solution to the puzzle, named "The Woman Freeman Will Marry", from [Summers 72]. A characteristic of this puzzle is that there is no straightforward path from the propositions in the puzzle's statement to the puzzle's solution. In solving this puzzle one has to raise hypotheses, reason from them and if a contradiction is detected replace some of those hypotheses and resume the reasoning. The statement of the puzzle is as follows:

> Freeman knows five women: Ada, Bea, Cyd, Deb and Eve. The women are in two age brackets: three women are under 30 and two women are over 30. Two women are teachers and the other three women are secretaries. Ada and Cyd are in the same age bracket. Deb and Eve are in different age brackets. Bea and Eve have the same occupation. Cyd and Deb have different occupations. Of the five women, Freeman will marry the teacher over 30. Who will Freeman marry?

Figure 1 shows the representation of every proposition in the puzzle's statement.[7] The wffs are described in a language called SNePSLOG [McKay and Martins 81] which is a logic programming interface to SNePS. Assertions and rules written in SNePSLOG are stored as structures in the SNePS network; SNePSLOG queries are translated into top-down deduction requests to the inference system; output from the inference is translated into SNePSLOG formulas for printing to the user.

In Figure 1 we represent the following propositions: There are five women, Ada, Bea, Cyd, Deb and Eve (wff1, wff2, wff3, wff4, wff5). Three women are under 30 (wff12)[8] and two women are over 30 (wff18). Every woman is either under 30 or over 30 (wff27).[9]

Two women are teachers (wff33) and the other three women are secretaries (wff39). The *the* in the previous sentence conveys the information that no woman is both a teacher and a secretary, represented by wff48. Ada and Cyd are in the same age bracket (wff53). Deb and Eve are in different age brackets (wff58). Bea and Eve have the same occupation (wff63). Cyd and Deb have different occupations (wff68). Exactly one woman over 30 is a teacher (wff79). Freeman will marry the teacher over 30 (wff88).

To solve the puzzle we raise hypotheses about the ages and professions of the women and ask SNeBR to deduce who Freeman will marry under those assumptions. If the hypotheses raised are consistent with the puzzle's statement the desired answer will be returned, otherwise a contradiction will be detected and SNeBR will guide us in discarding hypotheses.

[4] For example, exactly five out of ten propositions are true. Refer to the section on selecting between alternatives.

[5] We use the term *deduction rule* to refer to any proposition which has either a connective or a quantifier (or both). A deduction rule is a statement in the object language, and can be considered a recipe, plan or heuristic for deriving new information from old information.

[6] The multi-processing approach was influenced both by Kaplan's producer-consumer model [Kaplan 73] and by Wand's frame model of computation [Wand 74].

[7] The numbers associated with the wffs relate to the number of the node which represents the wff in the network.

[8] With this proposition we can see the advantage of the SNePS connectives. With the standard connectives this proposition would be expressed in the following way:
$(\neg age(Ada,u\text{-}30) \wedge \neg age(Bea,u\text{-}30) \wedge age(Cyd,u\text{-}30) \wedge age(Deb,u\text{-}30) \wedge age(Eve,u\text{-}30)) \vee$
$(\neg age(Ada,u\text{-}30) \wedge age(Bea,u\text{-}30) \wedge \neg age(Cyd,u\text{-}30) \wedge age(Deb,u\text{-}30) \wedge age(Eve,u\text{-}30)) \vee$
$\neg(age(Ada,u\text{-}30) \wedge age(Bea,u\text{-}30) \wedge age(Cyd,u\text{-}30) \wedge \neg age(Deb,u\text{-}30) \wedge age(Eve,u\text{-}30)) \vee$
$(\neg age(Ada,u\text{-}30) \wedge age(Bea,u\text{-}30) \wedge age(Cyd,u\text{-}30) \wedge age(Deb,u\text{-}30) \wedge \neg age(Eve,u\text{-}30)) \vee$
$(age(Ada,u\text{-}30) \wedge \neg age(Bea,u\text{-}30) \wedge \neg age(Cyd,u\text{-}30) \wedge age(Deb,u\text{-}30) \wedge age(Eve,u\text{-}30)) \vee$
$(age(Ada,u\text{-}30) \wedge \neg age(Bea,u\text{-}30) \wedge age(Cyd,u\text{-}30) \wedge \neg age(Deb,u\text{-}30) \wedge age(Eve,u\text{-}30)) \vee$
$(age(Ada,u\text{-}30) \wedge \neg age(Bea,u\text{-}30) \wedge age(Cyd,u\text{-}30) \wedge age(Deb,u\text{-}30) \wedge \neg age(Eve,u\text{-}30)) \vee$
$(age(Ada,u\text{-}30) \wedge age(Bea,u\text{-}30) \wedge \neg age(Cyd,u\text{-}30) \wedge \neg age(Deb,u\text{-}30) \wedge age(Eve,u\text{-}30)) \vee$
$(age(Ada,u\text{-}30) \wedge age(Bea,u\text{-}30) \wedge \neg age(Cyd,u\text{-}30) \wedge age(Deb,u\text{-}30) \wedge \neg age(Eve,u\text{-}30)) \vee$
$(age(Ada,u\text{-}30) \wedge age(Bea,u\text{-}30) \wedge age(Cyd,u\text{-}30) \wedge \neg age(Deb,u\text{-}30) \wedge \neg age(Eve,u\text{-}30))$

[9] information is implicitly contained in the statement of the puzzle.

wff1 : $Woman(Ada)$

wff2 : $Woman(Bea)$

wff3 : $Woman(Cyd)$

wff4 : $Woman(Deb)$

wff5 : $Woman(Eve)$

wff12 : ${}_5\alpha^3_3(age(Ada, u30), age(Bea, u30), age(Cyd, u30). age(Deb, u30), age(Eve, u30))$

wff18 : ${}_5\alpha^3_3(age(Ada, o30). age(Bea, o30), age(Cyd, o30). age(Deb, o30), age(Eve, o30))$

wff27 : $\forall\ (x)[Woman(x) \rightarrow {}_2\alpha^1_1\ (age(x, u30), age(x, o30))]$

wff33 : ${}_5\alpha^3_3(worker(Ada, teacher), worker(Bea, teacher). worker(Cyd, teacher), worker(Deb, teacher), worker(Eve, teacher))$

wff39 : ${}_5\alpha^3_3\ (worker(Eve, secretary). worker(Deb, secretary), worker(Cyd, secretary), worker(Bea, secretary), worker(Ada, secretary))$

wff48 : $\forall\ (x)[Woman(x) \rightarrow {}_2\alpha^1_1(worker(x, secretary), worker(x, teacher))]$

wff53 : $\forall\ (x)[{}_2\theta_1\ (age(Ada, x), age(Cyd, x))]$

wff58 : $\forall\ (x)[{}_2\alpha^1_1\ (age(Deb, x), age(Eve, x))]$

wff63 : $\forall\ (x)[{}_2\theta_1(worker(Bea, x), worker(Eve, x))]$

wff68 : $\forall\ (x)[{}_2\alpha^1_1\ (worker(Cyd, x), worker(Deb, x))]$

wff79 : ${}_5\alpha^3_3({}_2\alpha^2_2\ (age(Ada, o30), worker(Ada, teacher)),$
${}_2\alpha^2_2\ (age(Bea, o30), worker(Bea, teacher)),$
${}_2\alpha^2_2\ (age(Cyd, o30), worker(Cyd, teacher)),$
${}_2\alpha^2_2\ (age(Deb, o30), worker(Deb, teacher)),$
${}_2\alpha^2_2\ (age(Eve, o30), worker(Eve, teacher)))$

wff88 : $\forall\ (x)[{}_2\theta_1\ (marry(Freeman, x). {}_2\alpha^2_2(age(x, o30), worker(x, teacher)))]$

Figure .1: Propositions in the network

Using the propositions described in Figure 1, we built into the network the hypotheses represented in Figure 2. The hypothesis represented by wff6 states that there are five women and names those women, and the hypothesis represented by wff89 asserts all the specific information pertaining these women and their relationship with Freeman. The hypotheses represented by wff13, wff15, wff28, and wff31 define the ages and professions of the women.[10]

wff6 : ${}_5\alpha^5_5$ (wff5,wff4,wff3,wff2,wff1) hyp,{wff6} ,{}

wff89 : ${}_{12}\alpha^{12}_{12}$ (wff88, wff79, wff68, wff63, wff58,wff53, wff48, wff39, wff33,wff27, wff18, wff12) hyp,{wff89},{}

wff13 : $age(Ada, o30)$ | hyp,{wff13} ,{}

wff15 : $age(Cyd, o30)$ | hyp,{wff15} ,{}

wff28 : $worker(Ada, teacher)$ | hyp,{wff28} ,{}

wff31 : $worker(Deb, teacher)$ | hyp,{wff31} ,{}

Figure .2: Hypotheses raised

Suppose that we ask who Freeman will marry under the BS defined by the context {wff6, wff13, wff15, wff28, wff31, wff 89}. In this BS there is no assertion about who Freeman will marry but wff88 may enable its deduction. SNeBR sets up two sub-goals, finding who is over 30 and finding who is a teacher (Figure 3).

I wonder if $marry(Freeman, who)$
holds within the BS defined by the context (wff31 wff28 wff15 wff13 wff89 wff6)
let me try to use the rule $\forall(x)[{}_2\theta_1(x)(marry(Freeman, x), {}_2\alpha^2_2\ (age(x, o30), worker(x, teacher)))]$

I wonder if $age(x, o30)$
holds within the BS defined by the context (wff31 wff28 wff15 wff13 wff89 wff6)
I know $age(Cyd, o30)$

I know $age(Ada, o30)$
I wonder if $worker(x, teacher)$
holds within the BS defined by the context (wff31 wff28 wff15 wff13 wff89 wff6)
I know $worker(Deb, teacher)$
I know $worker(Ada, teacher)$
since $worker(Ada, teacher)$ and $age(Ada, o30)$ I infer $marry(Freeman, Ada)$

Figure .3: Ada and Cyd are over 30; Ada and Deb are teachers: Freeman will marry Ada

Figure 3. shows SNeBR's deduction that Freeman will marry Ada. The inference does not stop here, however, since there are several processes still waiting for answers and SNeBR reports inferences as shown in Figure 4.

since $age(Ada, o30)$ and $age(Cyd, o30)$
I infer ${}_1\alpha^0_0\ (age(Bea, o30))\ {}_1\alpha^0_0\ (age(Deb, o30))\ {}_1\alpha^0_0(age(Eve, o30))$
since not $age(Eve, o30)$ I infer $age(Deb, o30)$

Figure .4: Bea, Deb and Eve are not over 30 Deb is over 30

After the deduction of the information shown in Figure 4, a contradiction is detected (Figure 5). A contradiction will be detected by SNeBR when one of the following conditions occurs: (1) Nodes representing contradictory wffs are built into the BS under consideration;[11] (2) Information gathered by a connective elimination process shows that a rule is invalidated by the data in the BS.

In our example this latter case occurs: there exists one process to deduce information using the rule ${}_5\alpha^3_3(age(Ada, o30), age(Bea, o30), age(Cyd, o30), age(Deb, o30), age(Eve, o30))$ which gathers that there are three women who are over 30 (Ada, Cyd and Deb).

WARNING!
Contradiction detected in the following and-or ${}_5\alpha^3_3(age(Ada, o30)$
$age(Bea, o30), age(Cyd, o30). age(Deb, o30). age(Eve, o30))$
More true arguments than max. Arguments in wrong number $age(Ada, o30)age(Cyd, o30)age(Deb, o30)$ You have the following options.

1. Continue anyway, knowing that a contradiction is derivable;
2. Re-start the exact same request in a different context which is not inconsistent;
3. Drop the request altogether.

Do you want to continue anyway?
$\Rightarrow\Leftarrow$ n
Do you want to re-start the request in a new context?
$\Rightarrow\Leftarrow$ yes

Figure .5: A contradiction is detected

[10] Notice that specifying the ages of the two women over 30 completely determines the ages of the five women; and that specifying the names of the two women who are teachers completely determines the profession of the five women.

[11] If nodes representing contradictory propositions are built but one of them does not belong to the BS under consideration, SNeBR records that there is an inconsistent BS (which is not being considered) and proceeds. Refer to [Martins and Shapiro 83].

Upon detecting the contradiction SNeBR gives the options of continuing the reasoning within the inconsistent BS,[12] modifying the current context in order to obtain a consistent BS or giving up the request. In our example, we decided to restore consistency causing the interaction shown in Figures 6 and 7. [13]

Figure 6 shows the inspection of the hypotheses that are responsible for the contradiction. Although the context under consideration is the set {wff6, wff13, wff15, wff28, wff31, wff89} only the hypotheses represented by wff13, wff15 and wff89 were used in the derivation of the contradiction and thus they are the only ones whose change will restore consistency. The SWM system guarantees that removing *exactly* one of them will generate

In order to make the context consistent you must delete some hypotheses from the set (wff13 wff15 wff89) You are now entering a package that will enable you to delete some hypotheses from this set.
Do you want to take a look at wff13 ?
⇒⇐ n
There are 5 propositions depending on wff13 : (wff97 wff16 wff93 wff91 wff90)
Do you want to look at [a]ll of them, [s]ome of them, or [n]one?
⇒⇐ a
$_1\not\propto^0_5(marry(Freeman, Eve)) \mid ext, \{wff13, wff28, wff89\}, \{\{wff15\}\}$
.
.
What do you want to do with wff13 ?
[d]iscard from the context, [k]eep in the context, [u]ndecided, [q]uit this package
⇒⇐ d

Do you want to take a look at wff15 ?
⇒⇐ y
$age(Cyd, o30) \mid hyp.\{wff15\}, \{\{wff13, wff89\}\}$
There are 2 propositions depending on wff15 : (wff16 wff91) .
Do you want to look at [a]ll of them, [s]ome of them, or [n]one?
⇒⇐ n
What do you want to do with wff15 ?
[d]iscard from the context, [k]eep in the context, [u]ndecided, [q]uit this package
⇒⇐ d

Do you want to take a look at wff89 ?
⇒⇐ n
There are 8 propositions depending on wff89 :
(wff97 wff95 wff16 wff94 wff93 wff92 wff91 wff90) .
Do you want to look at [a]ll of them, [s]ome of them, or [n]one?
⇒⇐ n
What do you want to do with wff89 ?
[d]iscard from the context, [k]eep in the context, [u]ndecided, [q]uit this package
⇒⇐ k

Figure .6: Inspecting the inconsistent hypotheses

a context which is not known to be inconsistent. We keep the hypothesis concerning the statement of the puzzle (wff89) and discard the hypotheses concerning the women's ages (wff13 and wff15); We also enter new hypotheses concerning the women's ages (Figure 7).

After resolving the contradiction the inference resumes (Figure 8). In this case there is no further contradiction detected and SNeBR reports that Freeman will marry Deb and will not marry Ada, Bea, Cyd nor Eve.

[12] In SNeBR this is not dangerous since it is based on relevance logic in which the paradoxes of implication (e.g., from a contradiction anything can be derived) do not arise.

[13] Note that the restriction set of this extended wff has the set {wff15}, meaning that wff13, wff28, wff89, wff15 is a set kown to be inconsistent.

The following (not known to be inconsistent) set of hypotheses was also part of the context where the contradiction was derived: (wff31 wff28 wff6) Do you want to inspect or discard some of them?
⇒⇐ n

Do you want to add some new hypotheses?
⇒⇐ y
Enter an hypothesis using SNePSLOG
⇒⇐ $age(Bea, o30)$
Do you want to enter another hypothesis?
⇒⇐ y
Enter an hypothesis using SNePSLOG
⇒⇐ $age(Deb, o30)$
Do you want to enter another hypothesis?
⇒⇐ n

Figure .7: Adding new hypotheses

I wonder if $marry(Freeman, who)$
holds within the BS defined by the context (wff14 wff16 wff6 wff28 wff31 wff89)
I know $age(Deb, o30)$
I know $age(Bea, o30)$
I know $worker(Deb, teacher)$
I know $worker(Ada, teacher)$
since $_2\not\propto^2_2(age(Deb, o30), worker(Deb, teacher))$
I infer $marry(Freeman, Deb)$
since $age(Bea, o30)$ and $age(Deb, o30)$
I infer $_1\not\propto^0_0(age(Eve, o30))$ $_1\not\propto^0_0(age(Cyd, o30))$ $_1\not\propto^0_0(age(Ada, o30))$
since not $_2\not\propto^2_2(age(Eve, o30), worker(Eve, teacher))$
I infer $_1\not\propto^0_0(marry(Freeman, Eve))$

Figure .8: Freeman will marry Deb Eve,Cyd and Ada are not over 30 Freeman will not marry Eve

CONCLUDING REMARKS

We discussed SNeBR, the belief revision system used by SNePS; briefly described some of the concepts of the logic that underlies SNeBR; and showed an example. The example presented was obtained from an actual run just by slightly changing the syntax of the propositions.

SNeBR is implemented in SNePS, a powerfull knowledge representation system. A distinguishing characteristic of SNeBR is that it is based on a logic designed with the goal of supporting belief revision systems. SWM associates each proposition with all the hypotheses used in its derivation and with all the hypotheses with which it is known to be incompatible The SWM formalism guarantees that (1) The origin set of a supported wff contains every proposition that was used in its derivation. (2) The origin set of a supported wff only contains the hypotheses that were used in its derivation. (3) The restriction set of a supported wff records every set known to be inconsistent with the wff's origin set. (4) The application of the rules of inference is blocked if the resulting wff would have an origin set known to be inconsistent.

In SNeBR, propositions are represented by SNePS network nodes and are indexed by (linked with) the hypotheses in their origin set and the sets in their restriction set.

The queries to SNeBR are associated with a context, the network retrieval function only considers the propositions in the BS defined by that context. When a contradiction is detected, after selecting one hypothesis (or several hypotheses) as the culprit for the contradiction, the "removal" from the network of all the propositions depending on such hypothesis (hypotheses) is done just by dropping it (them) from the context being considered. Afterwards these propositions will no longer be in the BS under consideration and thus will not be considered by SNeBR.

ACKNOWLEDGEMENTS

Many thanks to Gerard Donlon, Donald McKay, Ernesto Morgado, Terry

Nutter, Bill Rapaport and the other members of the SNePS Research Group for their comments and criticisms concerning the current work.

REFERENCES

1. Anderson A. and Belnap N., **Entailment: The Logic of Relevance and Necessity**, Vol.1, Princeton University Press, 1975.

2. deKleer J., "Choices without Backtracking", **Proc. AAAI-84**, pp.79-85.

3. Doyle J., "A Truth Maintenance System", **Artificial Intelligence**, Vol.12, No.3, pp.231-272, 1979.

4. Kaplan R.M., "A Multi-processing Approach to Natural Language", **Proc. National Computer Conference 1973**, pp.435-440.

5. Martins J., "Reasoning in Multiple Belief Spaces", Ph.D. Dissertation, Department of Computer Science, SUNY at Buffalo, May 1983.

6. Martins J. and Shapiro S., "Reasoning in Multiple Belief Spaces", **Proc. IJCAI-83**, pp.370-373.

7. Martins J. and Shapiro S., "A Model for Belief Revision", **Proc. of the Non-Monotonic Reasoning Workshop**, pp.241-294, AAAI, 1984.

8. Martins J. and Shapiro S., "A Logic for Belief Revision", forthcoming.

9. McAllester D., "An Outlook on Truth Maintenance", A.I. Lab., M.I.T., AI Memo 551, 1980.

10. McKay D.P. and Martins J.P., "SNePSLOG User's Manual", SNeRG Technical Note No.4, Department of Computer Science, SUNY at Buffalo, 1981.

11. McKay D.P. and Shapiro S., "MULTI - A LISP Based Multiprocessing System", **Proc. 1980 LISP Conference**, pp.29-37.

12. McKay D.P. and Shapiro S., "Using Active Connection Graphs for Reasoning with Recursive Rules", **Proc. IJCAI-81**, pp.368-374.

13. Shapiro S., "The SNePS semantic network processing system", in **Associative Networks**, N.V.Findler (ed.), Academic Press, pp.179-203, 1979a.

14. Shapiro S., "Using Non-standard Connectives and Quantifiers for Representing Deduction Rules in a Semantic Network", presented at "Current Aspects of AI research", a seminar held at the Electrotechnical Laboratory, Tokyo, August 27-28, 1979b.

15. Shapiro S., Martins J.P. sand McKay D.P., "Bi-directional Inference", **Proc. of the Fourth Annual Conference of the Cognitive Science Society**, pp.90-93, 1982.

16. Shapiro S. and Wand M., "The Relevance of Relevance", Technical Report No.46, Computer Science Department, Indiana University, Indiana, 1976.

17. Steels L., "The Definition and Implementation of a Computer Programming Language based on Constraints", A.I. Lab., M.I.T., T TR-595, 1980.

18. Summers G., **Test your Logic**, Dover Publications, 1972.

19. Wand M., "The Frame Model of Computation", Technical Report N.20, Computer Science Department, Indiana University, Indiana, 1974.

GENIAL: UN GENERATEUR D'INTERFACE EN LANGUE NATURELLE

Bertrand Pelletier
Jean Vaucher*

Laboratoire Incognito
Département d'informatique et de recherche opérationnelle
Université de Montréal
C.P. 6128, Succursale A
Montréal H3C 3J7

Résumé -- GENIAL est un ensemble d'outils logiciels pour la construction rapide d'interfaces robustes en langue française. Le système, programmé en Prolog (environ 5000 lignes de code), est basé sur des techniques classiques: grammaire d'extraposition avec "slots" et transformation des requêtes en formules logiques. Le noyau linguistique renferme environ 90 règles syntaxiques et 130 mots utilitaires. Le système étant conçu pour le français, une attention toute particulière est portée à l'analyse morphologique. La réaction aux erreurs et les outils de génération sont aussi discutés en détail. Finalement, les résultats d'une première évaluation expérimentale d'une interface générée sont présentés.

Abstract -- GENIAL is a toolkit for the rapid construction of robust French natural language interfaces. The system, written in about 5000 lines of Prolog, is based on a combination of well-tried techniques: extraposition grammars, slots and translation of requests into logical formulae. The built-in linguistic core comprises about 90 syntatic rules and 130 common words. The paper describes the interface generation tools. Aspects of morphological analysis particular to French are also discussed in detail. Finally the results of a first experimental evaluation is given.

Introduction

Même si, pour des usagers expérimentés, l'emploi de langages artificiels permet une interaction rapide avec un débit élevé d'information; pour des débutants, la possibilité d'exprimer des requêtes en langue naturelle est un aspect important pour la convivialité d'un système.

Pour l'interrogation de bases de données, il existe des systèmes avec de bonnes interfaces en langue naturelle [8,9,11]. Ces systèmes sont cependant gros et hermétiques; de plus, il sont généralement orientés vers la langue anglaise.

Le système GENIAL que nous décrivons ici vient combler cette lacune. Il a été conçu pour expérimenter avec des interfaces en langue française dans des contextes très variés (systèmes experts, enseignement assisté par ordinateur, aide interactive, bureautique, etc.).

GENIAL répond aux besoins de deux clientèles: les informaticiens et les usagers. Les informaticiens cherchent un outil simple à comprendre et à utiliser pour générer et étendre des interfaces. Les usagers cherchent une interface "transparente", qui réagit vite et correctement, tolérant des fautes mineures, et instructive quand elle est incapable de comprendre une requête.

Dans ce qui suit, nous décrivons le fonctionnement de l'interface générée. Nous présentons les structures de données permettant efficacement la recherche dans le dictionnaire et la lemmatisation des mots, ainsi que le traitement des erreurs. Après, nous présentons les outils disponibles pour générer les interfaces. Finalement, les réactions d'un premier groupe d'usagers face à une interface générée sont données. Le lecteur désirant davantage de détails sur le système pourra consulter [5].

Description de l'interface

Nous décrivons dans cette section chacune des étapes du traitement, soit les analyses lexicale, morphologique, syntaxique et sémantique, puis la génération de la réponse.

Analyse lexicale

L'analyse lexicale est la toute première étape dans le traitement d'une requête formulée en langue naturelle. Le sens employé ici diffère de celui attribué en linguistique (analyse des termes en tant que lexèmes, etc.). Il s'agit du regroupement des caractères en unités lexicales, les mots ("tokens"), sans toutefois déterminer la catégorie lexicale de ces mots (verbe, nom, etc.).

Nous discuterons de différents problèmes rencontrés lors du découpage de la phrase en mots, notamment la transformation de caractères et le traitement des caractères spéciaux.

Lors de l'analyse lexicale, certaines transformations sont appliquées sur les caractères lus, afin d'obtenir la forme standard employée dans le dictionnaire. Par exemple, les lettres majuscules sont transformées en minuscules. De plus, les caractères accentués n'étant pas présents sur tous les terminaux, nous avons opté pour l'omission complète des accents, plutôt que pour l'emploi d'un symbolisme spécial (méta-caractère). L'exemple suivant illustre le rôle de caractères spéciaux (blanc, tiret, apostrophe) dans le découpage d'une phrase:

Aujourd'hui, l'élision est-elle encore un casse-tête ?

Nous traitons l'apostrophe comme un séparateur de mots. Cette convention conduit à certaines inconsistances, comme le démontre l'exemple précédent; il s'agit là, néanmoins, d'un phénomène extrêmement rare: moins de 10 occurrences figurent dans *Le Petit Robert* (*aujourd'hui*, *quelqu'un*, *n'importe qui*, etc.).

De même, nous considérons le tiret comme un séparateur. Bien que, là aussi, cette convention conduise à un découpage erroné dans certains cas (exemple précédent), il faut signaler que l'utilisation du tiret est très fréquente dans l'interrogation en langue naturelle (constructions interrogatives).

* présentement au Centre de Recherche Informatique de Montréal.

Un phénomène important qui doit être souligné au niveau de l'analyse lexicale est le traitement de la méta-langue. En effet, l'usager doit disposer d'un mécanisme permettant de distinguer les niveaux du langage dans une phrase telle que:

Connaissez-vous le mot mot ?

Cette distinction se fait dans l'interface par l'intermédiaire d'une calligraphie spéciale, soit l'encadrement du mot par des doubles apostrophes (*chaîne*):

Connaissez-vous le mot "mot" ?

Nous avons employé le formalisme des grammaires de clauses définies (DCG) [6] pour exprimer le traitement effectué lors de l'analyse lexicale. Comme nous le verrons dans la section suivante, l'analyse morphologique est grandement facilitée lorsque les mots sont représentés sous forme de listes de caractères plutôt que sous forme d'atomes (unités indivisibles). Par conséquent, l'analyse lexicale produit en sortie une liste de listes, chacune des sous-listes correspondant à un mot. Les sous-listes correspondant à une chaîne portent une marque spéciale:

[[c,o,n,n,a,i,s,s,e,z], [v,o,u,s], [l,e], [m,o,t], chaine([m,o,t])].

Analyse morphologique

L'analyse morphologique détermine la (les) catégorie(s) lexicale(s) de chacun des mots qui ont été découpés par l'analyse lexicale. Dans le cas d'un mot dont l'orthographe varie, il faut également déterminer la lemmatisation employée.

Dans ce qui suit, nous discuterons de la nécessité d'une bonne structuration des données minimisant à la fois la taille du dictionnaire et le temps de recherche. Nous introduirons finalement la structure résultant de l'analyse de la phrase: le mini-lexique.

Dans un système d'interrogation en langue naturelle qui se veut suffisamment robuste et qui doit par conséquent posséder une bonne couverture linguistique (environ 1000 mots; le français fondamental en contient 1700), le dictionnaire occupe certainement une grande partie de l'espace-mémoire. Il faut par conséquent une structure adéquate pour conserver cette masse d'information et y permettre un accès rapide.

Une première approche [4,8,11] consiste à inclure dans le dictionnaire une entrée pour chaque lemmatisation considérée d'un mot. Ceci peut être acceptable pour un dictionnaire anglais (peu de lemmatisation), mais non pour le français. Nous avons choisi d'incorporer au système des procédures de lemmatisation. Nous employons pour ce faire une méthode inspirée du programme de conjugaison proposé dans [2]. Notre approche est cependant plus simple d'utilisation tout en étant plus générale.

Nous avons défini le concept de "racine": il s'agit de la plus longue chaîne de caractères située au début du mot qui reste fixe indépendamment de la lemmatisation employée. Nous appellons "terminaison" du mot la chaîne de caractères située à la suite de la racine.

Par exemple, pour les mots "donnons", "professeurs", "auraient" et "des", les racines sont respectivement "donn", "professeur", "" (la chaîne vide) et "des", alors que les terminaisons sont "ons", "s", "auraient" et "". La figure suivante illustre les entrées du dictionnaire correspondant à ces mots:

```
dict([d,o,n,n|Term],          Term,   cat(verbe, er, donner)).
dict([p,r,o,f,e,s,s,e,u,r|Term], Term,  cat(nomcom, 1,
  (professeur, mas, A:prof:hum, [cdn(opt,de):B:matiere:nhum],
   professeur(X), X))).
dict([d,e,s],     [],     cat(det, _+plu, existe(X,P,1), X,_,P))).
dict([d,e,s],     [],     cat(prep, (des, nloc))).
dict([a,u,r|Term],        [a,u,r|Term], cat(verbe, avoir, avoir)).
```

Figure 1. Exemples d'entrées de dictionnaire.

Après avoir lancé, par exemple, le but "dict([d,o,n,n,o,n,s], Term, Info)", il s'agira de vérifier si la terminaison Term obtenue (ici [o,n,s]) est une lemmatisation valide, compte tenu de l'information Info (ici, conjugaison de type "er"), afin d'apporter des informations complémentaires sur la morphologie.

Afin de minimiser le temps d'accès au dictionnaire, nous employons une variation de la structure d'arbre TRIE, ne comprenant que 2 niveaux d'index (les 2 premiers caractères du mot). Pour en vérifier l'efficacité, nous avons effectué des tests sur un dictionnaire constitué des quelques 2600 mots de racines distinctes contenus dans *L'étranger*, de Camus. Le système utilisé était MProlog™ de Logicware roulant sur VAX 780.

Avec le dictionnaire implanté comme à la figure 1, sans structure de TRIE, le temps de calcul moyen pour accéder à un mot était de 1 seconde, ce qui n'en permet pas un emploi en temps réel. L'implantation en TRIE réduisait le temps de calcul par un facteur de quarante. De plus, la structuration en TRIE rend la performance du dictionnaire indépendante des techniques d'indexation utilisées par diverses implantations de Prolog.

L'échec est toujours possible lors de la recherche dans le dictionnaire, peu importe la taille de ce dernier (faute de frappe, mauvaise conjugaison, nouveau concept, synonyme, etc.). Lorsque ce type d'erreur se produit, l'interface pointe le mot fautif en indiquant qu'il est inconnu au dictionnaire, puis interrompt l'analyse de la requête. A ce stade, l'usager peut reformuler sa requête à l'aide des mécanismes d'édition incorporés.

Les sessions de travail des usagers sont mémorisées sur fichier, de même que la liste des mots inconnus au dictionnaire, avec leur fréquence d'emploi. Ces mémorisations servent d'une part à l'augmentation du corpus de phrases et de mots, et d'autre part à l'aide aux usagers (reprendre le déroulement d'une session).

Le résultat de l'analyse morphologique est un petit lexique qui contient toute l'information relative aux mots intervenant dans la requête. Il s'agit d'un mini-dictionnaire, limité aux mots de la phrase, qui permet d'éviter le retour en arrière ("backtracking") dans le dictionnaire entier ainsi que dans l'analyse morphologique. La grammaire accède à ce mini-lexique par une liste de positions dans la phrase ("[1,2,3,4]"), plutôt que par une liste de mots:

```
mini_lex(1, verbe,    (lister,ind+pre,   _+sin), [l,i,s,t,e]).
mini_lex(1, verbe,    (lister,imper+pre, _+sin), [l,i,s,t,e]).
mini_lex(1, verbe,    (lister,par+pas, mas+sin), [l,i,s,t,e]).
mini_lex(1, nomcom, (liste,mas+sin,A:liste:syste,
        [cdn(opt,de),_,_], liste(X), X), [l,i,s,t,e]).
mini_lex(2, prep,     (des,nloc), [d,e,s] ).
mini_lex(2, det,      (des,_+plu, existe(X,P,1), X,_,P), [d,e,s]).
mini_lex(3, nomcom, (professeur, mas+plu, A:prof:hum,
        [cdn(opt,de):B:matiere:nhum], professeur(X), X),
        [p,r,o,f,e,s,s,e,u,r,s]).
mini_lex(4, fin_de_phrase, rien, rien).
```

Figure 2. Mini-lexique pour la phrase " Liste des professeurs. " .

Analyse syntaxique

Il serait idéal de posséder un analyseur pour l'ensemble de la langue. Malheureusement, l'expérience démontre rapidement la difficulté, sinon l'impossibilité, de concevoir une grammaire complète du français. Dans ce qui suit, nous discuterons de divers critères qui influent sur l'élaboration de la grammaire utilisée.

D'abord, introduisons une terminologie qui aidera par la suite à différencier les niveaux de grammaire. Nous emploierons la notion de grammaticalité lorsqu'il sera question de la grammaire d'une langue naturelle, et de validité lorsqu'il s'agira de la description d'une grammaire dans un certain formalisme.

Ainsi, la grammaticalité d'une construction sera déterminée par la *compétence* alors qu'une phrase sera considérée valide relativement à un analyseur si elle est acceptée par cet analyseur. Pour terminer, mentionnons qu'aucune relation de cause à effet ne relie les concepts de grammaticalité et de validité. En particulier, nous justifierons dans les paragraphes suivants l'existence de règles valides mais non-grammaticales.

Elaboration d'une grammaire Un facteur important intervenant dans la détermination des règles de grammaire valides est le domaine d'application considéré. En effet, dans un univers fermé et bien maîtrisé (système de courrier électronique, traducteur automatique de bulletin météorologique), il est possible de former un corpus de phrases permettant la conception d'une grammaire puissante relativement au domaine traité.

Dans une interface en langue naturelle, il arrive fréquemment qu'un usager utilise des constructions non-grammaticales (ex: style télégraphique). Dans le cadre d'un système d'aide à l'apprentissage d'une nouvelle langue, il serait important d'ajouter les structures obtenues par la traduction mot à mot de la langue initiale [1].

De façon générale, même si elles présentent des déviations grammaticales, les requêtes soumises à l'interface ont un sens pour l'usager qui les fournit. Toute interface suffisamment robuste devrait donc pouvoir analyser un certain nombre de constructions non-grammaticales, prioritairement les plus courantes.

Formalisme de description Différents formalismes ont été présentés jusqu'à présent ([6,7,13]). Pour notre part, nous avons élaboré le système à partir d'une grammaire restreinte du français, construite par Alain Polguère du département de linguistique de l'Université de Montréal [10].

La grammaire, écrite en XG (grammaire d'extraposition) [7] reprend certaines idées sur l'emploi de l'extraposition à gauche dans les requêtes interrogatives ("Dans quelle ville ...") [8], idées auxquelles s'ajoutent de façon élégante les techniques de complémentation de [4]. Elle contient quelque 93 règles non-terminales et 16 règles terminales.

Nous y ajoutons l'idée d'une couche intermédiaire entre la grammaire et le dictionnaire (ici, le mini-lexique). Ceci nous permet de rendre la description de la grammaire indépendante de la structuration du dictionnaire et de retenir de l'information sur le déroulement de l'analyse.

Récupération d'échec Puisque le risque d'un échec lors de l'analyse syntaxique est toujours présent et ce, quelle que soit la complexité de la grammaire, il faut inclure dans une interface robuste un mécanisme de récupération d'erreur. La méthode privilégiée ici est l'heuristique du plus long chemin, employée dans différents systèmes robustes ([12]):

Heuristique 1:

> "Dans le cas d'un échec dans l'analyse de la phrase, la portion de l'analyse qui a permis de reconnaître la plus grande partie de la phrase est considérée comme le début correct de l'analyse".

L'emploi de cette heuristique est possible grâce à l'existence d'une couche intermédiaire entre la grammaire et le mini-lexique. En effet, lorsqu'un accès est effectué au mini-lexique en une position donnée de la phrase, l'interface retient la catégorie du mot cherché à cette position. Cette technique permet de donner, pour la requête "Quels sont les professeurs qui dans une université québécoise", le message:

> Désolé, il y a une erreur syntaxique dans votre requête.
> Quels sont les professeurs qui ...
> doit se poursuivre par l'une des catégories suivantes:
> verbe.

Le problème de l'accord est une cause d'erreur fréquente dans l'expression écrite. Pour éviter l'échec dans un tel cas, nous tentons une première analyse sous la contrainte de l'accord exact; s'il y a conflit entre deux syntagmes et qu'aucune analyse n'a encore été trouvée pour la phrase, une faute d'accord est notée et l'analyse se poursuit.

Analyse sémantique

Les arguments et prédicats intervenant dans les règles de grammaire se divisent en 2 groupes: ceux permettant la vérification de contraintes sémantiques (complémentation des verbes, noms, ... [4]) et ceux effectuant la construction de la formule logique.

Dans certains systèmes [4,8], seul le premier groupe intervient au niveau de la grammaire. L'arbre syntagmatique produit par l'analyse syntaxique est alors traité afin de construire une représentation sémantique de la requête.

Dans notre interface, les analyses syntaxique et sémantique s'effectuent en une seule passe (syntaxico-sémantique); cette passe fournit donc, en plus de l'arbre syntagmatique, une représentation sémantique de la requête. Pour distinguer les 2 types d'erreurs, nous avons employé avec un cerain succès l'hypothèse suivante

Heuristique 2 :

> "Lorsque l'analyse syntaxico-sémantique échoue, l'heuristique 1 est appliquée; si la catégorie lexicale du mot fautif obtenu ne figure pas dans la liste des catégories lexicales attendues, il s'agit d'une erreur syntaxique. Sinon, il s'agit d'une erreur sémantique."

Génération de la réponse

Lorsque toutes les étapes précédentes ont été franchies avec succès, la formule logique produite par l'analyse syntaxico-sémantique peut alors être évaluée. Dans ce qui suit, nous discuterons de la façon dont s'effectue cette évaluation.

L'interface produite par GENIAL possède 2 bases de données. L'une d'elles renferme certaines connaissances linguistiques de l'interface (vocabulaire) et permet de répondre à des requêtes telles que:

> Combien y a-t-il de verbes connus?
> Quels sont les noms communs ?

L'autre base contient les tuples relationnels concernant l'application considérée, de même que les prédicats d'accès à ces tuples. Par exemple, le prédicat "professeur(X)" peut être défini en fonction de la relation

"enseigner('Ducharme', 'ift-2030', 'aut-85')" par:

professeur(X) :- enseigner(X,_,_) .

Description du générateur d'interface

Nous présenterons ici le schéma général de GENIAL. Nous situerons également les tâches de l'implanteur, c'est-à-dire de celui dont le rôle est d'employer le système afin de générer une interface pour un domaine particulier.

La figure 3 présente le schéma du système. Nous y distinguons les fichiers constitués de programmes Prolog, les données fournies par l'implanteur (la partie ombrée correspond à la proportion des informations à ajouter aux données de base pour une nouvelle application) et les modules formant l'interface produite. Il y est notamment indiqué les étapes de traitement d'une requête, ainsi que les fichiers de messages d'aide à l'usager et de recueil cumulatif des sessions.

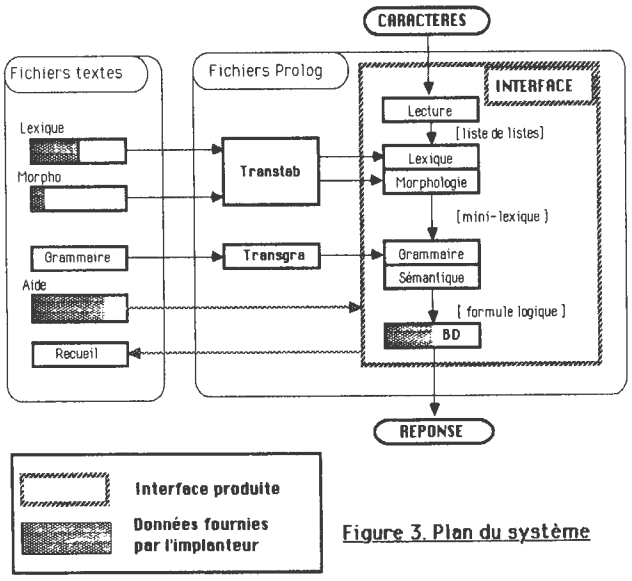


Figure 3. Plan du système

Une grammaire de base du français est fournie sous forme XG ainsi qu'un programme de transformation de cette grammaire en Prolog (Transgra), tiré de [7]. L'implanteur n'est donc pas dans l'obligation d'écrire une grammaire. De plus, l'outil de transformation en question lui permet de modifier à sa guise la grammaire fournie.

Le vocabulaire de base de l'interface et les terminaisons valides sont disponibles sous forme de "tables" dans les fichiers textes "Lexique" et "Morpho", comme l'illustre la figure suivante:

```
bloc( morphologie )   verbe  er
/* terminaison er comme "chanter" */

ind  pre      e   es  e   ons ez  ent
par  pas      e   es  ee  ees
inf  pre      er
```

Figure 4. Sous-ensemble de la morphologie des verbes.

Ces tables servent de modèle à l'implanteur pour l'ajout de son propre vocabulaire. La transformation de ces tables en fichiers Prolog est assurée par le programme "Transtab", lequel utilise des macros pour la génération du code Prolog. La base de données linguistique est crée automatiquement lors de cette transformation.

L'implanteur a également le rôle de fournir la base de données pour son application, de même que les prédicats d'accès à cette base. De plus, l'implanteur insère dans le fichier "aide" une description du schéma de sa base de données.

Evaluation de l'interface

Pour les fins de l'évaluation de GENIAL, nous avons créé une interface à une base de données contenant un sous-ensemble des prédicats et opérateurs MProlog™. L'interface produite a été expérimentée par un groupe d'usagers. Dans ce qui suit, nous introduirons la méthode employée pour l'évaluation de l'interface et nous enchaînerons par une discussion sur les réactions des usagers. Notre approche ne se veut nullement un protocole formel; une évaluation sérieuse aurait été souhaitable, mais débordait le cadre de notre travail.

Protocole d'évaluation

L'expérimentation a été effectuée avec 3 étudiants de l'université de Montréal qui avaient tous une bonne expérience dans l'emploi d'interfaces et qui connaissaient le domaine d'application: Martin Bourgault, Diane Goupil et Alain Polguère (l'auteur de la grammaire). Les sujets ont été mis à tour de rôle en contact avec l'interface. La séance était enregistrée sur bande sonore et les requêtes étaient conservées sur fichier.

La première phase de l'expérience, très courte, consistait en une présentation sommaire du système. Dans un second temps, le sujet était laissé à lui-même face au système, pendant 20 à 30 minutes. Dans la troisième étape, le sujet recevait de notre part la réponse à toutes les questions qu'il s'était posé jusqu'à ce moment. Lors de la quatrième étape, un certain nombre de questions étaient fournies à l'usager. Il devait les reformuler pour les soumettre au système.

Expérimentation

Dans un premier temps, nous relaterons les commentaires relatifs au domaine choisi pour l'expérimentation, soit un sous-ensemble des prédicats et opérateurs Prolog. Dans une seconde étape, nous résumerons les principales réactions des usagers devant l'interface comme telle.

Choix du domaine Notons premièrement que le fait de choisir comme domaine d'application, pour l'expérimentation, un sous-ensemble de Prolog, soit le langage dans lequel tout le système est écrit, a conduit à une certaine confusion chez les sujets.

Cette ambiguïté se situait notamment au niveau des opérateurs Prolog (and, +, ...): la définition de ces opérateurs en tant qu'objets dans la base de données était confondue avec la fonction des opérateurs des langages artificiels d'interrogation de banque de données.

Une autre source de confusion provenait du fait que les deux bases, en plus d'être écrites en Prolog, contenaient des informations se rapportant aux entités du langage Prolog. Par exemple, la méta-base permettait de répondre à une requête telle:

Est-ce que "predicat" est un nom commun?

Ici, "predicat" pouvait être considéré comme un nom commun faisant partie des connaissances linguistiques du système, ou encore comme une entité Prolog.

Réactions des usagers Notre première observation est l'emploi précoce de méta-questions. Citons par exemple:

Que puis-je faire ?
Je ne sais pas quoi faire.
Que savez-vous ?
A quoi sert votre système ?
Que contient la base ?

Face à l'échec de ces méta-questions, les sujets se tournaient vers l'aide disponible sous forme de menus. En effet, les messages d'erreurs, quoique significatifs, n'étaient pas pris en considération. De même, les sujets n'ont pas exploité les possibilités des heuristiques 1 et 2 pour construire mot-à-mot une requête valide.

A l'opposé, la découverte dans l'aide par menus d'une liste d'exemples de phrases valides a constitué une grande source d'inspiration: à partir de ce moment, les usagers ont su formuler correctement une requête faisant intervenir un nom propre, en encadrant ce dernier par des guillemets.

Tous les participants ont fait remarquer qu'ils auraient préféré avoir sur papier une copie de l'aide disponible interactivement, ou encore en posséder un résumé (5 à 10 pages).

Toutes les phrases entrées par les sujets étaient enregistrées sur un fichier de recueil; ce système nous a permis d'augmenter le vocabulaire et d'établir quelques statistiques sur l'expérimentation. Premièrement, notons qu'environ 47% des requêtes soumises (les requêtes excluent l'appel à l'aide par menus) ont été rejetées par l'interface, sur un total de 123 requêtes distinctes.

De plus, 3% des requêtes rejetées étaient agrammaticales, 69% étaient grammaticales et non-valides, 19% étaient grammaticales et "valides", mais faisaient intervenir des mots corrects mais inconnus (nouveau mot, nouveau concept), et 9% contenaient des mots erronés (faute de frappe, mot anglais).

Les méta-questions représentaient la majorité des requêtes grammaticales non-valides. Egalement, la convention lexicale sur l'entrée des noms propres a fait grimper les statistiques.

Finalement, remarquons que seulement 4% des requêtes soumises comportaient des fautes d'accord, lesquelles, comme indiqué précédemment, ne constituaient pas une cause d'échec.

Conclusion

Nous avons présenté GENIAL, un ensemble d'outils logiciels pour la construction rapide d'interfaces robustes en langue française.

L'analyse morphologique utilise des entrées lexicales sous forme de listes de caractères avec queues non-instanciées. Pour contrecarrer la lenteur inhérente à une telle structure, le lexique est implanté en arbre TRIE et les résultats sont gardés dans un mini-lexique qui sert de "cache" pour l'étape suivante. Des programmes spéciaux génèrent le dictionnaire sous la forme requise par le système.

Le système est tolérant aux fautes d'accord et une approche heuristique est employée pour émettre des diagnostics d'erreurs significatifs. Des informations sur les capacités linguistiques du système sont disponibles à l'usager.

Une première expérience avec le système a montré que la couverture linguistique de GENIAL était satisfaisante. Cependant, la carence de règles valides pour les méta-questions a engendré un fort taux d'échec: environ 40% des échecs relevaient des méta-connaissances. Finalement, nous avons noté une confusion sémantique due à la présence de deux bases de données distinctes.

Une plus grande intégration des méta-connaissances de même que l'incorporation de l'aide à l'usager sous forme de requêtes demeurent des problèmes intéressants à explorer.

Remerciements

GENIAL a été développé avec l'aide financière du Conseil de recherches en sciences naturelles et en génie du Canada.

L'expérimentation de l'interface à été effectuée avec la participation de 3 étudiants de l'Université de Montréal: Martin Bourgault, Diane Goupil et Alain Polguère.

Références

[1] P. J. Hayes et G. V. Mouradian, "Flexible Parsing", AJCL , (v.4-7, 1981), pp.232-242.

[2] H.Kanoui, PROLOG II: Manuel d'exemples , Groupe Intelligence Artificielle - E.R.A. C.N.R.S., Marseille, (1982).

[3] S. C. Kwasny et N. K. Sondheimer, "Relaxation Techniques for Parsing Grammatically Ill-Formed Input in Natural Language Understanding Systems", AJCL , (v.7-2, 1981), pp.99-108.

[4] M. C. McCord, "Using Slots and Modifiers in Logic Grammars for Natural Language", Artificial Intelligence , (v.18, 1982). pp.327-367.

[5] B. Pelletier, Système d'interrogation de banque de données en langue naturelle , Mémoire de maîtrise, Dept. d'informatique et de recherche opérationnelle, Université de Montréal, (1986).

[6] F. Pereira et D.H.D. Warren, "Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks", Artificial Intelligence , (v.13, 1980), pp.231-278.

[7] F. Pereira, "Extraposition Grammars", AJCL , (v.7-4, 1981), pp.243-256.

[8] F. Pereira, Logic for natural language analysis , Thèse de Ph. D., Université d'Edimbourg, aussi disponible comme Technical Note no 275, S.R.I. International, (1983).

[9] J.-F. Pique, Sur un modèle logique du langage naturel et son utilisation pour l'interrogation des banques de données , Thèse de 3e cycle, Université Aix-Marseille II, (1981).

[10] A. Polguère, Programmation Logique des Interfaces Langue Naturelle , Rapport de stage aux Laboratoires de Marcoussis C.R.C.G.E, (1984).

[11] D.H.D. Warren et F. Pereira, "An Efficient Easily Adaptable System for Interpreting Natural Language Queries", AJCL , (v.8-3, 1982), pp.110-122.

[12] R. M. Weischedel, "Responding Intelligently to Unparsable Inputs", AJCL , (v.6-2, 1980), pp.97-109.

[13] W. A. Woods, "Transition network grammars for natural language analysis", CACM , (v.13, 1970), pp.591-606.

TOWARDS A DOMAIN-INDEPENDENT METHOD OF COMPARING SEARCH ALGORITHM RUN-TIMES

H.W. Davis
R.B. Pollack
D.J. Golden

Wright State University
Dayton, Ohio

Abstract -- Reports in the literature comparing run-times of different search algorithms are highly dependent on parameter values of the domain where measurements are taken. Examples of such parameters are time required to expand a node and time required to calculate the heuristic distance between two problem states. Altering these parameter values can totally reverse the conclusion of a time comparison test. To eliminate this problem a new technique for comparing run-times of different search algorithms is presented. The iterative time-consuming activities of a search algorithm are parameterized. The values of these parameters are reported rather than raw timing data. The speed of two algorithms is now compared by examining their time-favorable regions in the corresponding parameter space. The effect is that time comparisons now depend only on the heuristic function used to measure the distances between two nodes and the structure of the search space graph. This, in turn, enables study of how the speed of different search algorithms varies with the heuristic-function/search-graph environment.

1. Introduction

This paper describes a technique for comparing the run-times of different search algorithms with one another on a set of problem instances. The goal is to report findings independent of machine speed or of such domain specifics as how long it takes to compute the heuristic distance between two problem states. In order to be useful to other workers, run-time comparisons should depend only on the state space graph and the heuristic functions used.

We have found empirically that the comparative run times of two search algorithms on the same problem instances can be reversed if one alters the ratio of time required to expand a node to time required to compute a heuristic distance. Thus, reporting the run times of two algorithms without adjusting for this ratio is meaningless. Yet, all comparisons in the literature do just this [1, 9].

It is also common to report the average number of node expansions as a measure of run-time [4, 7]. This is perfectly valid for some algorithms, such as unidirectional A*, provided comparisons are not to be made with algorithms which have different structures. Even here care must be taken. For example, in unidirectional A* implemented without hashing the number of node equality checks is proportional to the square of the number of node expansions. In certain problem domains the time for a node equality check exceeds node expansion time. Therefore using node expansion count as a measure of run-time can be misleading unless one knows the type of problem domain. If hashing is used, however, then node expansion time goes up and comparisons can only be made with other search algorithms that have a similar node expansion overhead. Another alternative is to have more than one measure of run-time, as is proposed below.

The approach described here is as follows: When reporting search algorithm run-times one lists the repetitive time consuming activities of the algorithm and associates with each such activity a time-complexity parameter. For a given problem domain and a given heuristic, or set of heuristics, a number of test runs are made. The average values of the parameters are reported in a time-complexity table (TCT). The TCT is domain independent in the sense that it depends only on the structure of the state space graph, the heuristic functions used and the search algorithm used. It does not depend on the amount of time required to do a single work unit associated with a time-complexity parameter or, more importantly, the ratio of such times. Another worker in a domain with a similar state space graph and similar heuristic power cannot generally use raw timing data from a different domain to help him pick a fast search algorithm. However, TCT data are sufficiently domain independent to be of value.

The TCT's are useful when comparing the run-times of two search algorithms. One sets up a 'parameter space' in which each axis corresponds to a single time-complexity parameter. From the TCT one discerns which region of the space is time-favorable to each algorithm. The result is what we call in section 3 a time-complexity graph. It is this region information, rather than timing data, that a worker in another domain needs to decide which search algorithm will be fastest for that domain. Time-complexity graphs give qualitative insight on the type of situations that make one search algorithm run faster than another. We are currently studying the extent to which they give predictive quantitative information.

In section 2 we describe time-complexity parameters using examples from five heuristic search algorithms which appear in the literature. In sections 3 and 4 we explain how these parameters are organized into a space whose regions correspond to situations in which one algorithm is faster than

another. An early version of these ideas appears in [2]. Section 5 illustrates the concepts of sections 3 and 4 by comparing the run times of three heuristic search algorithms.

2. Run-time usage in best-first search

We list below the major activities performed in heuristic search and indicate how this paper treats them. The activities vary depending on the algorithm and data structures used. We have tried to keep our methodology sufficiently general that its applicability to new algorithms and situations will be apparent. We use as examples five heuristic search algorithms found in the literature. The reader is assumed to be familiar with unidirectional A* (UA*) [5]. The other algorithms are bidirectional heuristic path algorithm (BHPA) [8], bidirectional heuristic front-to-front algorithm (BHFFA) [1], the d-node algorithm (DNODE) [9], and iterative-deepening A* (IDA*) [3, 4, 10]. These are summarized in an appendix.

Node expansion is performed once each cycle. Run-time depends on the problem domain and machine. X denotes the number of node expansions performed.

The heuristic distance between two nodes is calculated once for each non-duplicate successor in UA* and BHPA. In IDA* it is calculated once for each successor not duplicated on the active branch from start. It is performed more often in DNODE and BHFFA. H denotes the number of such calculations.

Each successor node is checked to see if it is a duplicate of a node already on the search tree (or trees, for bidirectional algorithms; active branch for IDA*). Duplicate checking can be very time consuming because problem states, in principle, can be arbitrarily complicated. N denotes the number of node equality checks.

There is an important situation in which N can be dropped from consideration, reducing the dimension of the parameter space and simplifying the analysis. This is if, in all the algorithms being compared, the whole pool of nodes to be checked for duplication is maintained in a hash structure. The effect is that we can absorb the time cost of node equality checks into node expansion time cost. This is because hashing time is O(1) and the number of hashes per cycle should approximate the average node degree of the search space graph. Hashing has been used for duplication checking in [8] and [9].

If the algorithms whose run-times are being compared include one for which hashing is not used, then N must be recorded for all the algorithms whether they hash or not; this is because absorbing hashing into node expansions gives the hashing algorithms a different node expansion time from the others and the analysis below would not be valid. In section 5 we describe another mechanism by which the N-dimension can be removed, but this involves a cost in domain independence.

OPEN is traversed looking for the minimum value of f, the evaluation function. This involves at each node a numeric comparison and possibly several assignment statements and arithmetic operations. The bidirectional algorithms usually do fewer such checks than UA*. IDA* does none. The number of nodes visited for this work is denoted M.

Maximum depth check comparison is done only in DNODE, and is done for each successor added to the search tree. It is denoted by D.

Threshold and minimum cost checks occur only in IDA*. The cost of each node generated is compared to the current cutoff threshold and to the current minimum value for setting the next threshold. T denotes the number of times this is done.

The discussion above isolates the following parameters as measures of run time usage:

- X : number of nodes expanded
- H : number of calculations of heuristic distance between two nodes
- N : number of node equality checks
- M : number of minimum f-value checks
- D : number of maximum depth checks
- T : number of threshold and minimum cost checks

In the following analysis we group the last three parameters into a single parameter,

$$W = M + D + T \qquad (1)$$

A more precise alternative is to carry M, D and T as separate parameters. The result is an analogous theory in which the 'critical parameter plane' is replaced by a hyperplane in a higher dimensional parameter space. We use (1) partly to simplify the theory described here and partly because in our example of section 5 only M and D are relevant. We have found that the time required by an M-work unit is within a few percent of the time required by a D-work unit in our implementations.

Major changes in the data structures used in a search algorithm may require that the resulting procedure be viewed as a different algorithm and that new parameters be introduced into the analysis. We have mentioned the example of finding duplicates in a hash versus non-hash structure.

We represent the run time of a search algorithm to a first approximation as

$$T = a_0X + a_1H + a_2W + a_3N \qquad (2)$$

The a_i are positive and depend on machine and problem domain. For example, in two problem domains with similar search space graphs and heuristic functions, the domain with a heuristic function taking longer to compute has the higher a_1 value. If two algorithms use the same heuristic distance, are applied to the same problem domain, and run on the same machine, then the corresponding a_i's are equal. Presumably the X, H, W, N-values would differ on identical problems. X, H, W and N are called time-complexity parameters.

3. Time-complexity graphs

Let A_i be a search algorithm with time-complexity parameters X_i, H_i, W_i, N_i, i = 1,2. From (2) we have

$$T(A_1) \leq T(A_2) \quad \text{iff} \tag{3}$$

$$\Delta X + (\Delta H)R_1 + (\Delta W)R_2 + (\Delta N)R_3 \leq 0 \tag{4}$$

where $\Delta X = X_1 - X_2$, $\Delta H = H_1 - H_2$, $\Delta W = W_1 - W_2$, $\Delta N = N_1 - N_2$, $R_1 = a_1/a_0$, $R_2 = a_2/a_0$, $R_3 = a_3/a_0$. It is important to note the physical interpretations of R_1, R_2, R_3 : they are, respectively, the average time requirements for an H, W, N work unit, normalized with respect to node expansion time. (4) says that the plane

$$\Delta X + (\Delta H)R_1 + (\Delta W)R_2 + (\Delta N)R_3 = 0 \tag{5}$$

divides the R_1, R_2, R_3 space into two regions which we call the A_i-favorable regions (i = 1,2). In one region the R_1, R_2, R_3 values are run-time favorable to A_1 and in the other they are favorable to A_2. We say that (5) represents a critical parameter plane and its graph in 3-dimensional space is a time-complexity graph.

In solving a search problem one normally has some flexibility regarding whether to use an accurate, time-consuming heuristic or a weaker, faster one. That is, a range of R_1 choices are normally available. There is no flexibility in R_2, R_3 values. The fastest algorithm for one choice of R_1 may not be the fastest for another; it depends on which side of the critical parameter plane the system is placed by the various R_1 values. Thus time-complexity graphs contain information which is critical for choosing the fastest algorithm for a given situation.

4. Comparing search algorithm run-times

Suppose we are testing run-times for search algorithms on a set of problem instances using one or more heuristics. A list of average values for time-complexity parameters (a TCT) is of more use than raw timing data. The TCT reflects where an algorithm spends its run time for a given search graph structure and group of heuristics, but it is not linked to the timing specifics of a particular problem domain.

To compare the run-times of two search algorithms in a given graph-heuristic environment, one combines their TCT's to form the corresponding time-complexity graph. Two types of uses for such graphs are as follows. (1) The qualitative features which make one algorithm more favorable than another are revealed. For example, if the critical parameter plane makes a steep slope with respect to the R_1 axis, then the amount of time for heuristic function calculation is a critical factor in choosing the fastest algorithm. A small slope means that it is not critical. (2) By determining R_1, R_2 R_3-values, a problem domain and the proposed heuristics single out a region of the parameter space. From a previously compiled time-complexity graph and this region information one can determine without tedious tests which search algorithm is fastest. We are currently studying the extent to which this may be done with precision. One question yet to be answered is to what extent do results obtained for one graph carry over to graphs with similar structures (eg. similar average branching factor, or density and size of cycles).

5. An example

To illustrate the ideas above we ran UA*, DNODE and BHPA on a sample of 15-puzzle problems. Each algorithm was run using an evaluation function of the form f = (1 - w)g + wh where w = 0.5, 0.75 and 1.0. h ranged over the four heuristics described in [9]. The 15-puzzle problems consisted of the same goal and ten different start states, each exactly 35 moves away from the goal. This provides 120 problem situations for each algorithm.

The sample set is small and is more illustrative of techniques than definitive. It has two desirable properties: (1) One can show by girth arguments that the diameter of the 15-puzzle problem space is in the low 70's. Thus 35 is probably in the (low) mid-range of 15-puzzle problem solutions. (2) Other researchers [1, 9] have terminated their program runs when X reached a predetermined number. They compare two algorithms with respect to some parameter by obtaining the average value of the parameter for a particular algorithm with respect to those problems solved by that algorithm. This is biased because each algorithm is only being evaluated on a problem set it finds easy; what's worse, this set may be relatively disjoint from the other's 'easy set.' To avoid this problem we were forced to drop four problem situations. Each algorithm was run to completion on the same 116 situations.

	X	H	N	W
UA*	4724	9528	222,017,977	51,980,044
BHPA	2028	4364	27,086,796	4,163,982
DNODE	1102	10135	7,208,565	1,136,008

Table 1. TCT: average values.

The three algorithms were run without hashing. The TCT is given in Table 1. We shall discuss the time-complexity graphs which go with Table 1, first assuming hashing and, second, without that assumption. We first discuss the qualitative and then the quantitative significance.

Had hashing been used, the same X, H and W values would have resulted but N would be 0 because node equality checking would be absorbed into node expansion, as explained in section 2. The time-complexity graphs are two-dimensional and are shown for our data in Figure 1. The time-favorable region is indicated for each of the three algorithms by placing a U, B or D on one side or the other of each of the three critical parameter lines. The locations of these regions reflect properties of the algorithms and give insight about when one is faster than another: The DB line has a higher

slope ($-\Delta H/\Delta W$) than the DU line because UA*'s single large search tree causes it to do more W-work than BHPA's two smaller trees. It crosses the R_1 axis closer to the origin ($-\Delta X/\Delta H$ = 0.16 versus 6.0) because BHPA expands fewer nodes and calculates fewer heuristic functions than does UA*, on average. The result is that BHPA is more favorable than UA* when compared to DNODE. For a fixed W-unit time and X-unit time, Figure 1 shows that both UA* and BHPA will run faster than DNODE if the time required to calculate the heuristic function is sufficiently large (ie., a large R_1 value). This is because, despite its virtues in expanding the fewest number of nodes, DNODE makes more heuristic function calculations than the other algorithms. The BU line indicates that BHPA is on average better than UA* for all R_1, R_2 values. This is because in our sample BHPA on average expands fewer nodes, evaluates fewer heuristic functions and does less W-work than UA*.

It should be emphasized that the qualitative conclusions we drew from Figure 1 are limited to search spaces with graph structure similar to that of the 15-puzzle and to heuristics whose effectiveness falls in the range of those we considered.

When hashing is not used, the time-complexity graph is three-dimensional. To simplify the analysis we use a simple devise to convert it to two dimensions. However, there is a tradeoff: the method instantiates the graph to certain timing aspects of our own problem domain. Another user would have to use Table 1 and his own problem domain to instantiate the situation differently. The method is as follows: We measure the average ratio of time to do an N-work unit to time required to do a W-work unit in our problem domain, say Z. (For the 15-puzzle at our location, Z = 0.97.) We absorb N into W by adding to the W-count Z times the N-count. The variable R_3 is now dropped from the critical parameter plane equation, yielding a line. Notice that it is not desirable to combine R_1 with R_2 or R_3 the way we just combined R_2 and R_3. As mentioned in section 3, R_1 is controlled by the designer of an artificial intelligence system when he chooses different heuristic functions. R_2 and R_3 are determined by the problem domain. By leaving R_1 free we allow the designer to see which algorithms and heuristic functions are fastest for his environment.

Figure 1 also shows critical parameter lines for our data assuming no hashing and adjusted in the fashion described above. The new lines have less slope (in absolute value), while their R_1 intercepts ($-X/-H$) are unchanged. The slope change in this case makes the less desirable algorithms even worse. From Table 1 we see the reason is that adding the N overhead to W will simply accentuate the present W-trend : UA* is worse than BHPA which is worse than DNODE. On the other hand, -H, -X are unchanged because failure to hash does not affect H, X.

We are currently studying the extent to which time-complexity graphs allow reliable quantitative prediction about which algorithms are fastest and by how much. This is as opposed to the qualitative

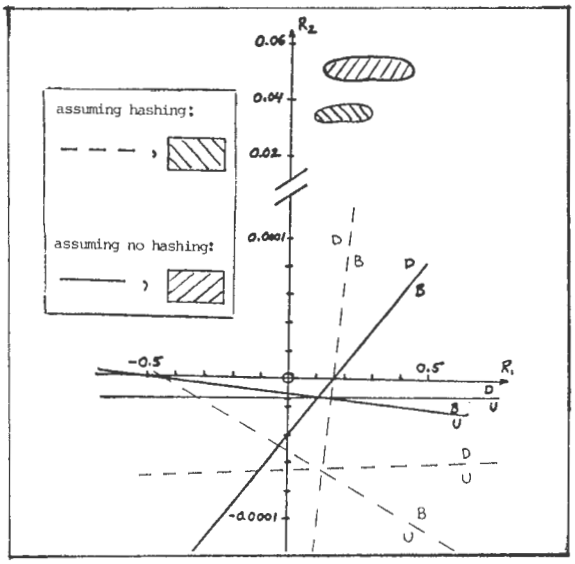


Figure 1. Time-complexity graphs.

	a_0	a_1	a_2	a_3
Hashing	1329	220	46.6	---
No Hashing	896	same	same	45.37

Table 2. Average a_i's in microseconds.

insight they provide about general circumstances which favor one or the other algorithm. Table 2 shows the average values we measured for the a_i's of equation (2). The shaded regions in Figure 1 shows the corresponding (R_1, R_2) regions relevant in our environment. According to the figure, the speed ranking of the algorithms is DNODE, BHPA and UA* when hashing is used. This is in essential agreement with results published in [9], but we have not checked it. The other shaded region indicates that without hashing the results are the same but more pronounced. Our UA*-normalized average time for DNODE and BHPA is 0.15 and 0.33, respectively. This is an accentuation of the results in [9].

6. Conclusions

We have described a method of reporting and comparing the run-times of search algorithms on a group of problem instances. The iterative time consuming activities of search algorithms are parameterized. Values of these parameters are reported rather than raw timing data. The effect is that the reported timing information now depends only on the algorithm, the heuristic function used, and the structure of the search space graph. The run-times of two algorithms are now compared by examining each algorithm's time-favorable region in a corresponding parameter space.

There are two advantages to this. (1) We may now study search algorithm run-time as a function of graph structure (average node degree, average number of cycles, etc.) and heuristic function accuracy (constant versus logarithmic relative

error, for example). Our results are independent of machine time and problem domain. (2) Workers can assess which algorithms are fastest for them by seeing the region of the parameter space into which their problem specifics fall. Data must first exist, however, for similar graphs and heuristic functions. The timing data currently being reported give no insight into which algorithms are fastest when the problem domain is altered.

Appendix

We assume the reader is familiar with unidirectional A* (UA*). Here we briefly describe the four other search algorithms discussed in the paper. The evaluation function is assumed to have the form f = g + h : g is the algorithm's current record of the least cost to a node from the root of the search tree on which it resides; h is the heuristic component of f. As in section 2, X and H denote, respectively, the number of node expansions made and the number of calculations of heuristic distance between two nodes.

Pohl's BHPA [8] is a generalization of UA* search in which two search trees are maintained, one rooted at the start node and the other at the goal. The h-value for each node is the heuristic distance to the root of the opposite tree. During each cycle of the algorithm, an open node is chosen for expansion from the tree with the smaller open list. The algorithm terminates when a node is chosen for expansion that appears on the opposite closed list, and there is no node on open with a smaller f-value.

A major problem with BHPA is that the search trees do not typically meet midway between start and goal. This results when the two trees develop different solution paths and meet near one of the endpoints. The effect limits the savings over UA* in X and H, at least in the 15-puzzle domain. Our own results and those of [9], however, show that BHPA finds 10% shorter path lengths than UA* (in the 15-puzzle domain).

De Champeaux and Sint's BHFFA [1] uses a dynamic heuristic to alleviate the problem of unbalanced search trees. The h-value of an open node is calculated in the following way: An estimated distance from the node to the opposite goal is computed as the heuristic distance from the node to an open node on the opposite search tree plus the g-value of that opposite node. Such an estimate is calculated using each node on the opposite open list and the minimum of these estimates is taken as the h-value. This results in a large increase in the number of heuristic distance calculations. In comparison with UA* H increases but X decreases. Our own experiments with the 15-puzzle show that, compared with UA*, BHPA and DNODE, BHFFA finds the highest quality solution paths (20% shorter than UA* on average).

Politowski and Pohl's d-node algorithm [9] chooses a node on each tree with the maximum g-value and designates it a 'd-node.' An h-value is the heuristic distance from a node to the opposite tree's d-node. This algorithm departs from the previously mentioned criterion for choosing direction: A fixed number of cycles execute from one direction, then the same is done from the other direction. At direction-change time heuristic distances normally need to be recalculated due to a changed d-node. Nevertheless DNODE's H values are considerably lower than BHFFA's. Its X values are intermediate between those of UA* and those of BHFFA. Unfortunately its solution quality appears low (50% longer than UA* on average in our experiments with the 15-puzzle).

Iterative-deepening A* (IDA*) works as follows [3, 4, 10] : Starting from the initial state, perform a depth-first search, cutting off a branch when its total cost (g + h) exceeds a given threshold. Repeat from start the depth-first search with successively increasing thresholds until a goal state is selected for expansion. The threshold starts at the heuristic estimate of the distance between start and goal. After each interation, the next threshold is the minimum of all cost-values that exceed the previous threshold.

References

[1] D. De Champeaux, and L. Sint, "An improved bi-directional heuristic search algorithm", Journal of the ACM,(1977), v.24,pp.177-191.

[2] H.W. Davis, R.B.Pollack, and D.J. Golden,, "A technique for comparing search algorithm run-times", Proceedings of the Fourteenth Annual ACM Computer Science Conference, (1986), pp.301-308.

[3] R. Korf, "Iterative-deepening-A* : an optimal admissible tree search", Proceedings of the 9th International Joint Conference on Artificial Intelligence, (1985), pp.1034-1036.

[4] R. Korf, "Depth-first iterative deepening: an optimal admissible tree search", Artificial Intelligence, (1985),pp.27:97-109.

[5] N. Nilsson, Problem Solving Methods in Artificial Intelligence,McGraw Hill,(1971), 255 pp.

[6] N. Nilsson, Principles of Artificial Intelligence,Tioga Publishing Co.,(1980), 476 pp.

[7] J. Pearl, Heuristics,Addison-Wesley Publishing Co., (1985), 382 pp.

[8] I. Pohl, Bi-directional and heuristic search in path problems, Stanford Linear Accelerator Center, SLAC Report No.104,(1969),157 pp.

[9] G. Politowski, and I. Pohl,"D-node retargeting in bidirectional heuristic search", Proceedings of the National Conference on Artificial Intelligence,(1984), pp.274-277.

[10] M. Stickel, and W. Tyson, "An analysis of consecutively bounded depth-first search with applications in automated deduction", Proceedings of the 9th International Joint Conference on Artificial Intelligence, (1985),pp.1073-1075.

PROPERTIES OF GREEDILY OPTIMIZED ORDERING PROBLEMS*

Rina Dechter and Avi Dechter**

Cognitive Systems Laboratory
Computer Science Department
University of California, Los Angeles, CA 90024

Abstract -- The greedy method is a well-known approach for problem solving directed mainly at the solution of optimization problems. The ability to characterize greedily optimized problems (i.e., that can be solved optimally by a greedy algorithm) is important for the mechanical discovery of hueristics and for the understanding of human problem solving. This paper discusses the properties of certain classes of greedily optimized ordering problems which are not covered under currently available theories (e.g., Matroids and Greedoids).

I. INTRODUCTION AND MOTIVATION

The greedy method is a well-known approach for problem solving directed mainly at the solution of optimization problems. Greedy algorithms use an irrevocable search control regime that uses local knowledge to construct a global solution in a ''hill climbing'' process [7]. Usually, they involve some real-valued function defined on the states of the search space. The greedy control strategy selects the next state so as to achieve the largest possible increase in the value of this function.

Our interest in greedy algorithms is twofold. First, **greedily optimized** problems (i.e., that can be solved optimally by a greedy algorithm) represent a class of relatively **easy** problems. Since many heuristics used in the solution of hard problems are related to simplified models of the problem domain [9], the ability to characterize easy problems is important, particularly if the process of discovering hueristics is to be mechanized. Second, greedy schemes are probably the closest to explaining human problem-solving strategies because they require only a minimum amount of memory space and because they often produce adequate results. (Due to the small size of human short-term memory, it is very hard to conceive of a human conducting best-first or even backtracking search, both requiring retention of some properties of previously suspended alternatives.)

Three examples of greedily optimized problems, along with their respective greedy strategies are given below:

1. **Minimum (Maximum) Spanning Tree.**
Given a graph $G = (V,E)$ where V is the set of vertices and E is the set of edges, and given a set of values associated with the edges, find a spanning tree with minimum sum of values.

   Greedy strategy: select edges in nondecreasing order of their values as long as they do not create a cycle.

2. **Job sequencing on a single processor.**
Given n jobs with processing time p_i and weight u_i for job i find a sequencing that will minimize the weighted mean flow time:

$$\bar{F} = \sum_{k=1}^{n} u_k \sum_{j=1}^{k} p_j ,$$

   where the jobs are indexed by their positions in the sequence.

   Greedy strategy : Sequence the jobs in a nondecreasing order of $\frac{p_i}{u_i}$.

3. **Optimal merge patterns.**
Merge n files in pairwise manner so that the total merging cost will be minimized. Each file has weight w_i. The cost of merging files i and k into a new file j is:

$$w_j = w_i + w_k .$$

   Greedy strategy: Merge the two files which have the lowest cost, then add the resultant file to the list and repeat (Huffman procedure).

All three problems involve the task of selecting in some order, from a given set of elements (edges of a graph, jobs, files), a subset (not necessarily proper) that satisfy some property, so as to maximize (or minimize) the value of a cost function defined on all possible solutions. These problems demonstrate a useful classification of greedily optimized problems. The first is an example of a **selection problem**, where the cost function is not dependent on the order of the elements in the subset of elements which constitutes a solution. Other examples of greedily optimized selection problems are the continuous knapsack problem and the problem of storing programs on a limited amount of tape. The second is an **ordering problem**, where the value of the cost function is dependent on the order of the elements as well as on their identity. Other examples in this class are scheduling sequential search

*This work was supported in part by the National Science Foundation, Grant #DCR 85-01234

**R. Dechter is also with the Artificial Intelligence Center, Hughes Aircraft Company. A. Dechter is also with the California State University, Northridge, CA

[10] and minimizing maximum flow-time in a two machine flow-shop [1]. The third problem is an instance of a **tree construction problem**. These problems require the generation of a tree which induces a partial order on the set of elements.

The solution of selection and tree-construction problems by greedy algorithms has been studied extensively. Specifically, selection problems are covered by the work of Edmonds and Gale [6] on the relationships between matroid theory and greedy algorithm. Tree-construction problems are summarized by the work of Parker [8] which characterizes the set of cost functions defined on trees that are minimized using the Huffman algorithm (which is a greedy procedure).

No comparable body of knowledge exists for characterizing greedily optimized ordering problems. A generalization of the Matroid structure, called Greedoids [4], represents an attempt in this direction. However, many greedily optimized ordering problems (including the job sequencing problem mentioned above) cannot be patterned after either matroid or greedoid theories. In this paper we present a theory intended to fill this gap by characterizing cost functions that permit the optimal solution of ordering problems by a greedy algorithm.

The remainder of the paper is organized as follows. In section 2 we introduce the notion of a Problem Scheme, used to describe classes of ordering problems, and state the necessary and sufficient conditions for a problem scheme to be greedily optimized. Section 3 discusses a special class of greedily optimized problems, those which have uniform ranking functions. In section 4 we focus on a particular type of greedily optimized problems having a dominant solution and provide some necessary and sufficient conditions for problems to be in this class. The various conditions and their possible uses are illustrated in section 5 with the aid of two examples. A summary and conclusions are given in Section 6. The theorems in this paper are given without proofs. The proofs can be found in [2].

II. PRELIMINARIES: DEFINITIONS AND NOMENCLATURE

Most, if not all, ordering problems can be described in terms of a **Problem Scheme** P defined as a triplet (E,PAR,C) where:

1. E is a set of elements.
2. PAR is a set of parameters, which are real valued functions defined over the elements of E. Parameters could be single-argument functions, in which case they are denoted by $\alpha(i)$, $\beta(i)$, etc., where i indexes the elements in E. They could also be multiple-argument functions, defined over all sequences of the same number of elements and denoted by $\gamma(i,j)$, $\delta(i,j,k)$, etc.
3. C is a real valued cost function. It associates a cost with any sequence of subsets of elements in E, and is dependent only on the the parameters of the elements and on their order. The cost function is written as $C = C(\sigma)$, where σ denotes a sequence of elements in E, and σ_i is the element in position i in σ.

A problem scheme describes an entire class, or family, of problems that are very similar in character. An instance of a problem scheme P, denoted by P_I is specified by a subset of elements E_I of E along with the values of their parameters. Let n be the cardinality of E_I. The problem associated with every instance P_I of P is to find a sequence $(e_1, \cdots, e_n)$ of all the elements of E_I such that $C(e_1,\ldots,e_n)$ is maximal (or minimal) over all permutations of the elements.

For example, the problem of sequencing jobs on a single processor so as to minimize a weighted average of the flow times can be formulated in terms of a scheme (E,PAR,C) where E is a set of jobs, PAR contain two parameters, p and u, that associate with each job a processing time and an importance weight, respectively, and C is a cost function defined on any sequence σ of n elements as follows:

$$C(\sigma) = \sum_{i=1}^{n} u_i \sum_{j=1}^{i} p_j \tag{1}$$

where u_i and p_i are the parameters of the i^{th} element in the sequence σ.

Definition: A **greedy rule** for P is a sequence of functions $f = \{f_j\}$

$$f_j : \Phi_j \rightarrow R \tag{2}$$

where Φ_j are all sequences of j elements.

A greedy procedure for solving any instance P_I of P using $f = \{f_i\}$ is defined as follows (maximization is assumed here and hereafter):

Greedy(P,f):

1. choose $e \in E$ such that $f_1(e) = \max\{f_1(e') \mid e' \in E\}$
2. after choosing $e_1, e_2, \ldots, e_{i-1}$ choose $y \in E - \{e_1, \ldots, e_{i-1}\}$ such that $f_i(e_1, \ldots, e_{i-1}, y) = \max\{f_i(e_1, \ldots, e_{i-1}, x) \mid x \in E - \{e_1, \ldots, e_{i-1}\}\}$

Definition: P is **greedily optimized** (by f) if for every problem instance P_I having n elements, the sequence $(e_1, \ldots, e_n)$ generated by Greedy(P,f) has a maximum cost over all permutations of the elements.

In the balance of this paper we restrict our attention to problems with single-argument parameters and (unless specified otherwise) to greedy rules that satisfy

$$f_i(e_1, \ldots, e_i) = f_i(e_i) = f(e_i) . \tag{3}$$

In that case the greedy rule is defined over the set of parameters associated with each element, i.e.,

$$f(e_i) = f(\alpha_i, \beta_i, \ldots) . \tag{4}$$

We will refer to such rules as **ranking functions**. Possible ranking functions for the job sequencing problem discussed above are:

$$f(u_i, p_i) = u_i \tag{5}$$

and

$$f(u_i, p_i) = p_i u_i . \tag{6}$$

A ranking function induces a weak order among the elements of E and the greedy procedure simply chooses elements in a nonincreasing order of f.

Definition: A ranking function is **optimizing** for some problem scheme P, if for every problem instance, P_I, it generates an optimal order. A problem scheme is said to be **greedily optimized** if it permits an optimizing ranking function.

We now give a necessary and sufficient condition for a problem scheme to be greedily optimized. The condition is stated for problems having a one-to-one cost function (i.e., no two sequences have the same cost). It should be slightly modified to hold for any cost function but this involves some details that we choose to avoid for the sake of simplicity.

Theorem 1:
A necessary and sufficient condition for a problem scheme P, having a one-to-one cost function, to be greedily optimized is that for any two elements a and b (characterized by their assigned parameters) and for all problem instances of P in which they both participate, either a precedes b in all optimal sequences or b precedes a in all optimal sequences.

□

Consider, for example, a problem scheme defined by a set of four elements $\{1,2,3,4\}$ and some cost function. Suppose that the optimal sequence for the instance defined by the set $\{1,2,3\}$ is (3,2,1), and that the optimal sequence for the instance defined by the set $\{1,2,4\}$ is (4,1,2). Then, this schema does not have any optimizing ranking function because elements 1 and 2 do not have the same ordering in the two optimal sequences.

The verification of the condition given in Theorem 1 usually depends on the knowledge of the optimal solution of the problem, and cannot be carried out by simple manipulation of the problem representation. Therefore, its usefulness is very limited. In the next section we present stronger, potentially more easily verifiable, requirements that constitute sufficient (but not necessary) conditions for greedily optimized problems.

III. UNIFORM RANKING FUNCTIONS

A reasonable approach for obtaining sufficient conditions for greedily optimizing problems is to extend the properties required by Theorem 1 for the optimal sequence to all possible sequences. This leads to the following definition.

Definition: Let $P=(E,PAR,C)$ and let f be a ranking function (not necessarily one-to-one) defined over the set of parameters of each element in E. A ranking function f is called **uniform** relative to P, if for every problem instance and for every sequence σ of elements in that problem instance,

$$C(\sigma) \geq C(\sigma^i) \text{ if } f(\sigma_i) > f(\sigma_{i+1}) \qquad (7)$$

and

$$C(\sigma) = C(\sigma^i) \text{ if } f(\sigma_i) = f(\sigma_{i+1})$$

for all i, where σ^i is the sequence resulting from the exchange of the i^{th} and $i+1^{th}$ elements in σ.

Theorem 2:
A problem scheme P that has a uniform ranking function f, is greedily optimized by it.

□

We next address the question of the properties the cost function should have to guarantee the existence of a uniform ranking function. Let Σ_{ab} be the set of all the sequences, in all the instances of problem scheme P, for which element a immediately precedes element b.

Definition: A cost function C is said to be **pairwise preferentially independent** (p.w.p.i.) if $\forall\ a, b$ either

$$C(\sigma) \geq C(\sigma^a) \quad \forall\ \sigma \in \Sigma_{ab}$$

with strict inequality for at least one sequence, or

$$C(\sigma) \leq C(\sigma^a) \quad \forall\ \sigma \in \Sigma_{ab}\ , \qquad (8)$$

with strict inequality for at least one sequence, or

$$C(\sigma) = C(\sigma^a) \quad \forall\ \sigma \in \Sigma_{ab}\ ,$$

where σ^a is the sequence resulting by the exchange of the adjacent elements a and b in σ. In the first two cases we say that C prefers a on b (resp. b on a), and denote it by $a >_{p.w.} b$ (resp. $b >_{p.w.} a$). In the third case we say that C is indifferent between a and b and use the notation $a \sim_{p.w.} b$.

definition: A pairwise preferentially independent cost function C is said to be acyclic if the relation $\geq_{p.w.}$ is transitive, i.e.,

$$\text{if } a \geq_{p.w.} b \text{ and } b \geq_{p.w.} c \text{ then } a \geq_{p.w.} c\ . \qquad (9)$$

This last property (i.e., that the relation "C prefers a on b" satisfies transitivity) is required to assure a weak order [5] and does not follow automatically from p.w.p.i. The following example shows that a cost function can be pairwise preferentially independent but not acyclic. Consider a problem instance defined by a set of three elements $\{1,2,3\}$ with a cost function C that creates the following complete order among all different sequences:

$$(321)>(132)>(213)>(312)>(123)>(231)\ .$$

For this instance, C is pairwise preferentially independent where 3 is preferred to 2 and 2 is preferred to 1. However, 1 is preferred to 3 thus violating transitivity.

Theorem 3:
A necessary and sufficient condition for a problem scheme $P = (E,PAR,C)$ to have a uniform ranking function is that C is p.w.p.i. and acyclic.

□

In the remainder of this paper we use the term p.w.p.i to include both properties. The next theorem suggests a possible process for identifying a p.w.p.i. cost function and for discovering its optimizing ranking function.

Theorem 4:
Let P be a problem scheme $P=(E,PAR,C)$ and σ any sequence of any subset of the elements in E. If the cost function C satisfies

$$C(\sigma) - C(\sigma^i) = K \cdot (d(\sigma_i) - d(\sigma_{i+1})) \qquad (10)$$

for all i, where K is a nonnegative function defined on σ and d is a function defined on the parameters associated with each element, then d is an optimizing ranking function.

□

The process Theorem 4 suggests is: perform symbolic manipulation on the cost function and try to express the difference between the cost of an arbitrary sequence and that of the sequence which results from exchanging the i^{th} and $i+1^{th}$ elements. If the expression satisfies condition (10) then an optimizing ranking function is given by d in that expression.

As an example consider again the single-processor job sequencing problem whose cost function is given in (1). Let σ^i be a sequence resulting from the exchange of the i^{th} and $i+1^{th}$ elements. We get that

$$C(\sigma) - C(\sigma^i) = (u_{i+1}p_i - u_i p_{i+1}) = u_{i+1}u_i \left(\frac{p_i}{u_i} - \frac{p_{i+1}}{u_{i+1}}\right). \qquad (11)$$

In this case the ranking function suggested from the above representation is $f(p_i,u_i)=\frac{p_i}{u_i}$.

The gap between the sufficient condition for a problem scheme to be greedily optimized, namely, that the cost function be p.w.p.i., and the necessary condition as given in Theorem 1, is not as wide as it may seem. For instance, in problem instances of two elements the two conditions coincide, because there are only two possible sequences, one of which has maximum cost. This observation leads to another way for discovering optimizing ranking functions for any greedily optimized problem (not necessarily with a p.w.p.i. cost function).

Theorem 5:
If P is any greedily optimized problem scheme then a ranking function f is optimizing if and only if it agrees with the ordering dictated by the cost function on pairs of elements, that is, for every two elements a and b, if $C(a,b) > C(b,a)$ then $f(a) > f(b)$.

□

The proof of Theorem 5 relies on our understanding that the ranking function is optimal for all instances of a problem scheme and in particular for instances of two elements. Practically all optimizing ranking functions for problems known to be greedily optimized satisfy this property.

Theorem 5 suggests that an optimal solution to a greedily optimized ordering problem is obtained by simply sorting the elements in the order dictated by applying the cost function to pairs of elements. When a problem is not known a' priori to be greedily optimized, this method could be used to either generate candidate ranking functions or to reject the hypothesis that the problem is greedily optimized (if, for instance, the pair-wise preference turns out to be non-transitive). When the objective is to find an explicit optimizing ranking function f then, by Theorem 5, candidate functions must satisfy $C(a,b)>C(b,a)\rightarrow f(a)>f(b)$ for any two elements a and b of E.

The above observations imply that cost functions defined on pairs of elements can be used as building blocks for generating cost functions that are greedily optimized. A cost function C_2, defined on pairs of elements with k parameters each, is said to be **transitive** if for every $x,y,z \in R^k$

$$C_2(x,y)\geq C_2(y,x) \text{ and } C_2(y,z)\geq C_2(z,y) \rightarrow C_2(x,z)\geq C_2(z,x) \quad (12)$$

Theorem 6:
If a cost function C_2 defined on pairs is transitive, then a problem scheme $P=(E,PAR,C)$, where C is given by:

$$C(e_1,e_2,\ldots,e_n)=\sum_{j=1}^{n}\sum_{k=1}^{j}C_2(e_k,e_j) \quad (13)$$

is greedily optimized.

□

IV. DOMINANT RANKING FUNCTIONS

A common greedy rule is the cost function itself, i.e.,

$$f_i(e_1,\ldots,e_i)=C(e_1,\ldots,e_i)\,. \quad (14)$$

This means that at each step the algorithm chooses that element which, if it was the last, would yield best cost (i.e., Myopic policy). The Greedoid Theory [4] is concerned solely with this greedy rule.

Greedy rule (14) is optimal for some problem scheme P if any instance P_I of P has the following property: any subsequence $(e_1,\ldots,e_j)$ of E_I that has a maximal cost over all subsets of size j of E_I can be extended to a sequence of length $j+1$ that has a maximal cost over all subsets of size $j+1$ of E_I. Formally,

$$\forall (e_1,\ldots,e_i) \text{ optimal over } \Phi_i \;\; \exists \; e_{i+1}\in E_I-\{e_1,\ldots,e_i\} \quad (15)$$

$$\text{such that } (e_1,\ldots,e_i,e_{i+1}) \text{ is optimal on } \Phi_{i+1}\,.$$

When this condition is satisfied, the greedy rule (14) generates an optimal sequence, σ, satisfying the following property: any subsequence $(e_1,\ldots,e_j)$ of σ has a maximal cost over all subsets of size j of E_I. An optimal sequence that has this property is called a **dominant sequence**. A greedy rule that generates dominant sequences for every problem instance is said to be **dominant**. A problem scheme (or its cost function) that has a dominant greedy rule is said to be **dominantly optimized**. It is clear then, that a problem scheme that satisfies condition (15) is dominantly optimized and its dominant sequences can be obtained using the greedy rule (14).

Dominance is not a necessary condition for the optimality of a greedy rule. For example, the cost function

$$C(\sigma)=\sum_{i=1}^{n}u_i\sum_{j=1}^{i}p_j \quad (16)$$

which, as we already know, is greedily optimized by the ranking function $f(u,p)=\frac{p}{u}$, is not dominant. To see this, consider the three-element problem instance, in which each element i is defined by its parameters (u_i, p_i):

$$e_1=(1,4),\;\; e_2=(0.5,3)\;\; e_3=(5,10) \quad (17)$$

The values assigned by the ranking function to the elements are: $f(e_1)=4$, $f(e_2)=6$, $f(e_3)=2$, so that the optimal sequence is: (e_2,e_1,e_3). Evaluating sequences of two elements we see that

$$C(e_2,e_1)=8.5 \quad (18)$$

while

$$C(e_3,e_1)=105 \quad (19)$$

Obviously, the length-2 subsequence of the optimal sequence is not maximal, and thus the ranking function is not dominant.

We now identify necessary and sufficient conditions for a cost function to be dominantly optimized and discuss the relationships between these conditions and those obtained for non-dominant greedily optimized problems. We are interested in identifying sufficient conditions which are stronger than (14) and may be easier to verify. Again, we focus on greedy rules which are ranking functions.

Since a dominant ranking function is optimizing, it is determined by the order imposed by the cost function on pairs of elements and should satisfy the condition of Theorem 5, and since the cost function is defined for sequences of one element, we must have $f(e)=C(e)$ and:

$$\text{if } C(a,b) > C(b,a) \text{ then } C(a) > C(b)\,. \quad (20)$$

To test this property, one should check the behavior of $C(e)$ as a ranking function: if inconsistency is found in the order induced by $C(e)$ and the order induced by the cost on pairs

(given that both orders are well defined) then the hypothesis that the problem has a dominant optimizing ranking function can be rejected. (It still may have a non-dominant optimizing ranking function as in (17).)

The necessary conditions for a dominant cost function requires the following definitions. Let Σ^t_{ab} denote all sequences for which element a immediately precedes b when b is the last element in the sequence.

Definition: A cost function is **tail pairwise preferentially independent** (t.p.w.p.i.) if either

$$C(\sigma) \geq C(\sigma^a) \quad \forall \quad \sigma \in \Sigma^t_{ab} \tag{21}$$

with strict inequality for at least one sequence, or

$$C(\sigma) \leq C(\sigma^a) \quad \forall \quad \sigma \in \Sigma^t_{ab}$$

with strict inequality for at least one sequence, or

$$C(\sigma) = C(\sigma^a) \quad \forall \quad \sigma \in \Sigma^t_{ab}$$

In the first two cases we say that C prefers a on b (resp. b on a) t.p.w., and denote it by $a >_{t.p.w.} b$ (resp. $b >_{t.p.w.} a$). In the third case we say that C is indifferent between a and b t.p.w and use the notation $a \sim_{t.p.w.} b$. If the relation is acyclic then it induces a weak order on the elements of E.

Definition: A cost function is **order preserving** if

$$\forall\, i \text{ if } C(e_1,\ldots,e_i) \geq C(e_1',\ldots,e_i') \text{ and } C(e_{i+1}) \geq C(e_{i+1}') \tag{22}$$
$$\text{then } C(e_1,\ldots,e_i,e_{i+1}) \geq C(e_1',\ldots,e_i',e_{i+1}').$$

Theorem 7:
Let C be a cost function in $P = (E, PAR, C)$ which is both order preserving and t.p.w.p.i. If $C(a) \geq C(b)$ implies that $C(a,b) \geq C(b,a)$, then C is greedily optimized by a dominant ranking function.

□

It is easy to show that a cost function which is t.p.w.p.i. and order preserving is also p.w.p.i. Therefore, a cost function satisfying the conditions of Theorem 7 is p.w.p.i. and has a dominant optimizing ranking function. For example, the function

$$\sum_{i=1}^{n} p_1 \cdot p_2 \cdot \ldots \cdot p_i \tag{23}$$

is both p.w.p.i. and order preserving. The ranking function $f(p) = p$ results in a dominant optimal solution. For C(e) to be an optimizing ranking function, it is important to verify that the order induced by $C(e)$ agrees with that induced by the p.w.p.i. property (i.e. the last condition of Theorem 7). For instance, the cost function

$$\sum_{i=1}^{n} {p_i}^i \tag{24}$$

is both p.w.p.i. and order preserving. However, the ordering implied by $C(p)$ results in a decreasing sequence while the (optimal) ordering implied by exchanging adjacent elements is nondecreasing in p. Indeed, this cost function is greedily optimized with $f(p) = -p$, but is not dominantly optimized.

Up to now all the "nice" greedily optimized cost function properties we described required the cost function to be p.w.p.i. (or t.p.w.p.i.). Since this property is not a necessary condition it is natural to look for cost functions that are greedily optimized but not p.w.p.i. We show that the p.w.p.i. property may indeed be replaced by another strong property of cost functions.

Definition: A cost function is **strong order preserving** if $\forall i$ and $\forall\, x, y \in E$

$$C(e_1,\ldots,e_i) > C(e'_1,\ldots,e'_i) \rightarrow \tag{25}$$
$$C(e_1,\ldots,e_i,x) > C(e'_1,\ldots,e'_i,y)$$

For example, A lexicographical order among sequence of integers is strong order preserving. The cost function

$$C(e_1,\ldots,e_n) = \sum_{i=1}^{n} |e_i - e_{i+1}| \, 10^{n-i} \tag{26}$$

for $e_i \in [0,10)$, is strong order preserving but not p.w.p.i.

Theorem 8:
If a cost function is strong order preserving then it is dominantly optimized.

□

V. EXAMPLES

In this section we illustrate the ideas presented above by verifying the properties of two greedily optimized problems.

The first problem is **Sequencing Jobs according to due-dates.** Given n jobs, each associated with a deadline d_i and a processing time p_i, find an optimal sequencing that minimized the maximum job lateness defined by:

$$\max\{F_i - d_i\}$$

Where F_i is the flow time of job i defined by:

$$F_i = \sum_{j=1}^{i} p_j$$

Jackson [3] had shown that the problem is p.w.p.i. and suggested the due-dates as the ranking function.

Let $e_1 = (p_1, d_1)$ and $e_2 = (p_2, d_2)$ be a pair of jobs with their processing times and due-dates. Using the process suggested by Theorem 5, of identifying ranking functions based on costs of pairs, it can be shown that

$$C(e_1,e_2) > C(e_2,e_1) \rightarrow d_1 > d_2, \tag{27}$$

that is,

$$\max\{p_1 - d_1, p_1 + p_2 - d_2\} > \max\{p_2 - d_2, p_2 + p_1 - d_1\} \tag{28}$$
$$\rightarrow d_1 > d_2.$$

The cost of one element is

$$C(e_1) = p_1 - d_1. \tag{29}$$

This cost does not provide the same ordering as the due-dates and, therefore, the cost function is greedily optimized but not dominant.

The second problem is **Minimizing maximum Flow-time in a two-machine flow-shop.** Let (A_i, B_i) be a pair associated with each job. A_i is the work to perform on the first machine of the shop and B_i is the work to be performed on the second machine. For each i A_i must be completed before B_i can begin. Given the 2n values: $A_1, A_2, \ldots, A_n$, $B_1, B_2, \ldots, B_n$, find the ordering of these jobs on each of the two machines so that neither the precedence nor the occupancy constraints are violated and so that the maximum of the flow time F_i is made as small as possible.

It has been shown [1] that the maximum flow-time is minimized if job j precedes job $j+1$ when

$$\min\{A_j,B_{j+1}\} < \min\{A_{j+1},B_j\} . \quad (30)$$

Looking only on two-job problems, it is easily verified that the ordering dictated by (30) coincides with the order determined by costs on pairs. If (A_1,B_1) and (A_2,B_2) are two jobs then the cost function for the sequence (e_1,e_2) is:

$$C(e_1,e_2) = A_1 + \max\{A_2,B_1\} + B_2 . \quad (31)$$

It can be shown that

$$A_1 + \max\{A_2,B_1\} + B_2 \; < \; A_2 + \max\{A_1,B_2\} + B_1 \quad (32)$$

iff

$$\min\{A_1,B_2\} < \min\{A_2,B_1\} \quad (33)$$

This criterion is the one known in the literature. From the transitivity property of the order induced by (33) we know that there exist a ranking function f that induces an **individual** order. After some manipulation such a ranking function can indeed be formed. It can be shown that if:

$$\min\{A_1,B_2\} < \min\{A_2,B_1\} \quad (34)$$

then

$$\frac{sign(A_1 - B_1)}{\min(A_1 , B_1)} < \frac{sign(A_2 - B_2)}{\min(A_2 , B_2)} . \quad (35)$$

Therefore the function

$$f(A_i,B_i) = \frac{sign(A_i - B_i)}{\min(A_i , B_i)} \quad (36)$$

is a uniform ranking function for the problem. The cost for one element only is $A_1 + B_1$ and it does not coincide with the ranking function (36). We can therefore conclude that this cost function is not dominant.

VI. SUMMARY AND CONCLUSIONS

This paper discusses the conditions for optimization ordering problems to be greedily optimized and the relationships between these conditions and the corresponding optimizing greedy rules. Within the greedily optimized problems, we identify a class of pairwise preferentially independent and acyclic (p.w.p.i) problems that are optimized via a uniform ranking function, and show that several well known examples fall into this class. We provide a procedure which, in certain cases, can be used to verify the p.w.p.i property and to identify uniform ranking functions.

Observing that the optimal solutions to many greedily optimized selection and tree-construction problems are **dominant**, we study this property in connection with ordering problems. This property is of particular interest because (as human intuition usually dictates) it makes the cost function itself an optimizing greedy rule. We show that p.w.p.i problems do not necessarily have dominant solutions, and that dominantly optimized problems are not necessarily p.w.p.i. We also give sufficient conditions for a problem to be both p.w.p.i and dominantly optimized. The relationship between the different classes is presented in Figure 1.

ACKNOWLEDGEMENT

We would like to thank Judea Pearl for contributing some of the ideas developed in this paper.

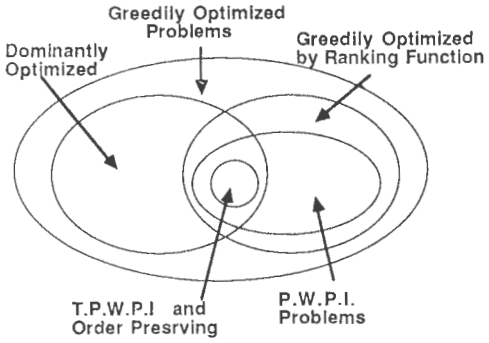


Figure 1 - Relationships Among Classes of Greedily Optimized Problems

REFERENCES

[1] Conway, R.W., W.L. Maxwell, and L.W. Miller, *Theory of scheduling*, Reading, Massachusetts: Addison-wesley Publishing company, 1967.

[2] Dechter, R., "Properties of greedily optimized problems," UCLA, Summer quartely , Los Angeles, Cal, Tech. Rep. Computer Science department, 1985.

[3] Jackson, J. R., "Scheduling a production line to minimize maximum tardiness," UCLA, Los Angeles, Cal, Tech. Rep. Management Sciences Research Project, 1955.

[4] Korte, B. and L. Lovasz, "Mathematical structures underlying greedy algorithms.," in *Proceedings Fundamentals of computation theory,* Szeged, Hungary: 1981, pp. 205-210. Springer Verlang's lecture Notes in Computer-Science #117.

[5] Krantz, D., D. Luce, S. Suppes, and A. Tversky, *Foundations of Measurement, Vol 1.*, New-York and London: Academic Press, 1971.

[6] Lawler, E.L., *Combinatorial optimization, Networks and Matroids*: Holt, Rinehart, and Winston, 1976.

[7] Nilsson, N., *Principals of Artificial Intelligence*, Palo Alto, California: Tioga, 1980.

[8] Parker, D.S., "Conditions for optimality of the Huffman algorithm.," *SIAM Journal of Computing*, Vol. 9, No. 3, 1980.

[9] Pearl, J., "On the discovery and generation of certain heuristics," *AI Magazine,* No. 22-23, 1983.

[10] Simon, H. A. and J. B. Kadane, "Optimal problem solving search: all or none solutions.," *Artificaial Intelligence,* Vol. 6, 1975, pp. 235-247.

Mechanisms in ISFI; A Technical Overview
(Short Form)

Gary A. Cleveland

The MITRE Corporation, Burlington Road
Bedford, MA 01730

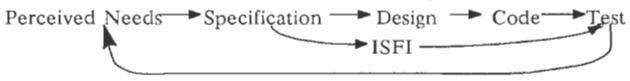
This paper is intended as a technical overview of the automatic programming system ISFI in its first three years. This paper deals with how necessary and desirable qualities and capabilities appear in ISFI. In many cases, ISFI borrows from earlier work, extends and customizes the work in some way, and then coordinates it in one cohesive system. However, it is the belief of the designers of ISFI that the system has something to offer as an exploratory vehicle both in terms of what can be accomplished with existing technologies (such as scope, reasoning power, level of interaction, etc.) and in developing new technologies where needed.

Software Crisis

In the late seventies, DoD was spending $4 billion a year on software. Current estimates are that in the early nineties this figure will rise to around $40 billion, a ten-fold increase in as many years. More generally, the need for software and software engineers in the world at large is growing exponentially, but productivity is only growing at about five percent per year ([Frenkel85]). Confound these problems with a shortage of knowledgeable people and one has the primary ingredients for the well-publicized "software crisis." This paper presents a technical overview of an experimental solution to the software crisis.

ISFI is a theory and implementation of knowledge-based automatic programming that produces high-level source code from a specification of behavior. For this reason, the "software production cycle" can be short-circuited and the way that software is produced can be fundamentally changed.

Software Production Cycle



ISFI is intended to assist a user in creating a specification (in a "network of constraints" formalism) and then design and code the problem automatically. This paper examines the mechanisms used to achieve this performance in the first three years.

What is Automatic Programming?

In the view of the designers and implementers of ISFI, an automatic programming system is one that accepts as input a non-procedural specification of behavior and produces as output a program that exhibits the specified behavior. Simply put, the eventual user of such a system should be able to state what he wants to happen (without necessarily telling how it should happen – that's programming) and be given a program that makes it happen.

This research was sponsored by Rome Air Force Development Center under Contract F19628-86-C-0001, project 7310.

In an object oriented world, behavior of a program takes two forms: 1) the input/output behavior and 2) the non-monotonic behavior (i.e., side effects). In the first form, some input objects are used to compute some output objects (the output objects are created according to the relationships given by the program specification; see [Brown85A] for philosophical discussion). In the second form, the set of relationships among the objects is changed by a program. For our model, this change can always be described by adding (deleting) an object to (from) some property, part, or attribute of a given object. These two forms of behavior are not mutually exclusive; a program can both perform side effects and compute objects as outputs.

An automatic programming system should be able to exhibit a number of kinds of behavior including forward-chained deduction, reasoning with incomplete or inconsistent specifications, automated design of algorithms and data structures, procedural and data abstraction, and explanation of its behavior. Obviously, an automatic programming system must do much more than this but these are some of the capabilities that have been addressed in the first three years.

What is ISFI?

ISFI is a knowledge-based system designed for intelligent automation of the programming process. ISFI takes an object oriented view of the world and transforms relationships among objects into computations that perform the actions implicit in the relationships. While other systems have been developed to explore many of the mechanisms that ISFI uses, ISFI seems to be the first of its kind as a comprehensive, problem solving, language independent, automatic programming system that operates from a non-procedural specification. Many systems explore various issues of program synthesis by transformations ([Balzer83], [Barstow83], [Boyle84], [Brown81]) non-monotonicity ([Doyle78], [McDermott80]) , propagation through semantic nets ([Brown81]), problem solving ([Davis80]), code generation for compilers ([Aho79], [Steele78]), and knowledge-based optimizations ([Green81A+B], [Rich81], [Kant83]) but none seems to carry through to the entire programming process. It is the intention of ISFI's designers to address the software production cycle in such a way that a software system can be debugged or altered by making changes to the specification rather than to the program. In other words, a user of ISFI would observe the behavior of a system and make his changes in the specification rather than in the code.

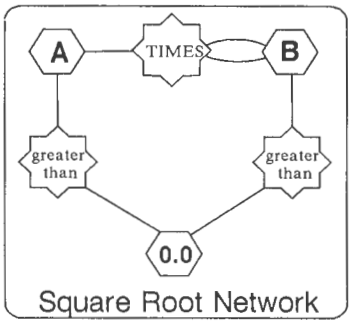
A point to emphasize is that ISFI is not tied to any given target language. The representations of objects, their behavior, and the synthesized program are all free of language dependent

considerations. Only a very small section of ISFI's code generator actually deals with the target language's naming conventions, scoping, parameter passing, and other considerations.

ISFI'S Objects.

ISFI maintains a straightforward representation of objects and their relationships in a network of constraints. Objects are represented as nodes while their relationships are represented by constraints. Two objects are related if their nodes are connected through some set of constraints. A special kind of relationship representing parts, attributes, and properties of objects appears in ISFI as **role-descriptors**. An important distinction between ISFI's role-descriptors and the attributes, slots, parts, etc. that are found in most other knowledge-based systems is that role-descriptors combine in a single mechanism the notions of parts, attributes, slots, etc, whereas most other systems tend to treat these separately (see [Brachman83], [Swartout83], [Martin79] [Minsky75]). Another important distinction is that role-descriptors may contain either an object or a collection of objects. Collections allow the intelligent cooperation between data structures that hold multiple objects of the same type, and real world objects that have multiple-valued attributes. Further, we can now index and order this collection when it is implemented as a data structure that supports indexing and ordering. Reasoning about collections in these ways is a natural part of ISFI's task.

One of the most interesting claims we make about ISFI's networks of constraints is that they express a program specification in a non-procedural form. That is, the networks are not programs themselves. A good example of this is a specification to find square roots of real numbers. The network that expresses this behavior is shown below. The important point here is that no part of the network defines the algorithm to be used in the synthesized program. The specification simply states that B times B equals A and that both A and B are greater than zero.



Mechanisms in ISFI.

ISFI utilizes a number of mechanisms in program synthesis. A synopsis of these mechanisms follows. Among these mechanisms is the planner/problem-solver that invokes other mechanisms according to current goals.

Propagation.

Propagation, ISFI's primary mechanism for dealing with networks of constraints, attempts to produce computations for objects in some nodes given computations for objects in other nodes. Propagation always starts at some set of nodes (such as inputs and constants) that have computations (also called **values**) and attempts to find computations for other nodes until the process can go no further. Each step of propagation through the network is accomplished by **law application**, wherein the computational interpretations of the constraints in a network (called **laws**) are used to compute some nodes connected to the constraint based on others. Thus, law application provides individual steps and propagation chains those steps together to walk through an entire network. Each law that successfully computes the values of some new node queues up the new node for further propagation. When the queue empties, ISFI is confident that all computations in the network that can be found have been found.

One of the interesting things about propagation is that it can fail. Propagation can only string together known computations derived from the constraints in the network. If no computation chain can be found, propagation will fail and the planner/problem-solver must be notified to invoke a mechanism such as inheritance or transformations to introduce such a chain into the network. In a sense, propagation finds the easy answers, an important task.

Inheritance.

In order to make use of generic facts, ISFI copies the nodes and constraints from the networks in the domain knowledge base into the networks where programs are being synthesized. Knowledge copied in this way is said to be inherited. Through this mechanism we can declare facts such as, "POSITIVE-INTEGERS are greater than zero." and expect them to be utilized appropriately. Note that this sort of statement can be used in many ways in many contexts. The declaration does not determine the usage.

Further note that these declarations are not assumptions and as such can not be overridden. These "for-all" or class declarations must be true for every object in a particular class (or we must be willing to ignore the exceptions). Assumptions that can be retracted or overridden are handled by an entirely different mechanism. Thus, inheritance means something different in ISFI than it does in some other systems such as FRL. (See [FRL77] for details and [Brachman83] for a discussion of types of inheritance.)

Inheritance in a network of constraints consists of copying network structure (nodes and constraints) from one place to another. Several things complicate this process. First, ISFI must not form duplicate structure to what is already present. That is, if some portion of a network is already represented, it should not be added again. Secondly, **doublets** may be introduced by copying network structure and if so they must be found and eliminated. A doublet is a type of configuration of nodes and constraints in the network that allows one to deduce equality between some objects (see [Brown81] for more details). Eliminating doublets amounts to explicitly expressing this implied equality. Lastly, when we inherit class information about a particular node, we may want to inherit information from the superior classes (in the class hierarchy) as well. Thus, inheritance needs to track up the hierarchy to inherit all relevant information.

Transformations.

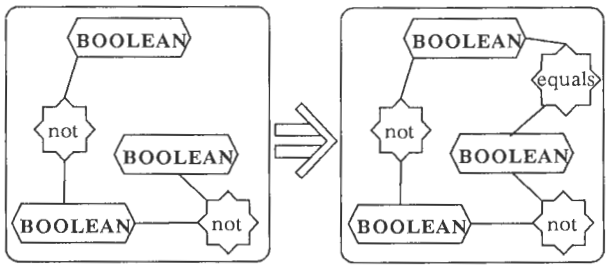
Transformations, like inheritance, are used to copy domain knowledge in the form of networks from one place to another. Transformations, though, are a form of inference rule and have two parts which correspond to the hypothesis and conclusion. The hypothesis is a pattern network that is matched against the network in which the rule is being applied. If the match is successful then the conclusion (target) network is copied into the original network. This copying is done in exactly the same way that the copying is done for the inheritance mechanism. The problem of combinatorial explosion in applying transformations is addressed by

the planning/problem-solving mechanism. ISFI can plan what transformations to apply because each transformation has an indication of what goals its inference is likely to achieve.

Transformations are the key mechanism for converting a non-procedural specification to a procedural specification - in other words, finding an algorithm that matches the original specification. This is equivalent to intorducing a computation path into the network. A transformation may, for example, introduce a binary search computation (formulated in a group of nodes and constraints) where the value of some node is known on an interval. After this transformation has been applied, the network would contain an algorithmic description that can be used to derive a computaion for the desired node. Transformations are also used to convert existing algorithms to other, more efficient algorithms. An important distinction is that while networks of contraints may contain algorithmic descriptions, they need not. Networks are not necessarily a procedural specification (can not, in general, be considered a program) and must be considered non-procedural.

An example of a transformation is shown below. This transformation does not introduce a computation path where none existed previously, but it does introduce a computation path that is less expensive than the path already known. This transformation performs the classic optimization of eliminating double negatives in boolean expressions.

Double Negative Transformation



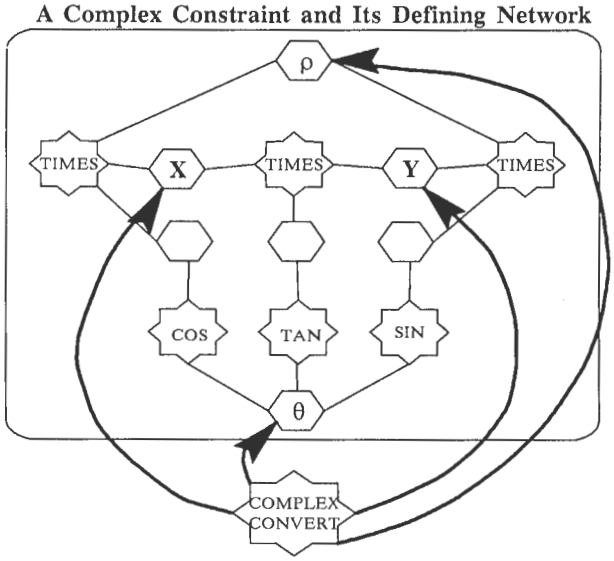
pattern network **target network**

Use of Complex Constraints.
Recall that a constraint expresses computational relationships among the nodes that it is connected to. In complex constraints, these relationships are expressed in a separate network of constraints. The complex constraint can then use the network in a number of ways to help solve the current programming problem. Thus, a single constraint is used to embody an entire network of knowledge with the result that ISFI has access to a form of procedural abstraction.

Complex constraints may be used in several ways, based on the results desired. First, the laws of a constraint may be matched to computational paths through the constraints defining network. If these paths correspond to the computation we desire to use, then the laws can be used in exactly the same manner as user-defined laws with their computational interpretations. Secondly, the network of a constraint may be copied into the network that is using the complex constraint. This is the same sort of copying as discussed for inheritance and transformations. After copying, the added network structure is in no way special and is used in the same ways as any other portion of network.

An example of a complex constraint and its defining network is shown below. This example could be used to express the conversion from (X,Y) (rectangular) coordinates to (ρ,θ) (polar) coordinates with a single constraint.

A Complex Constraint and Its Defining Network



Mutations and Consequence Management.
Side effects in ISFI are represented as **mutations** and are discussed in detail in [Brown85B]. A mutation is defined by the node being mutated, the role-descriptor, the node that represents the contents of the role, and the operation type (insert or delete depending on whether some object is being added to or taken away from the role). This implies that, while ISFI must deal with the changes made by a program, the kinds of changes that can be made are limited to changes in the attributes (roles) of objects. This limitation is one of notation, not one of representational power. We merely state that the only relationship that can change is the role membership of some object.

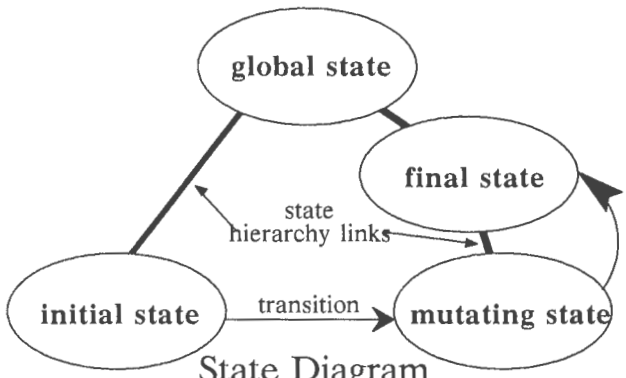
Of course, the hard problems involving side effects have always lain in managing the consequences and timing of those side effects. Managing the consequences is classically known as the "frame problem". Till now the best solution to this problem lay in "reason maintenance" ([Doyle79], [McDermott83]) but that solution could not be easily applied because it depends upon knowing the identity of the objects being discussed. As mentioned previously, ISFI needs to deal with objects whose identity will not be revealed until the synthesized program is executed. A comparison that might be made is that while reason maintenance interprets a changing facts and their consequences, ISFI must compile such descriptions for later execution.

In ISFI, the frame problem may be stated as, "Given some changes to the properties (roles) of some objects, what changes to the properties of other objects need to be made to maintain consistency?" (Consistency is specified by some set of constraints.) Timing the side effects, at least as far as a partial ordering, calls for some sort of state mechanism where each state represents some context or situation. Situations change by traversing from one state to another.

In ISFI, the problem of managing consequences boils down to collecting the necessary set of constraints into the problem network and making sure that they get used in the correct way. While normally all constraints are interpreted as holding true, in this case ISFI collects a set of constraints that must be made true. Making a constraint become true is accomplished by mutations. In other words, a constraint is forced to be true by declaring an appropriate mutation. This "appropriate" mutation is derived based on the laws of the constraint type. In order to collect the constraints that must be enforced, ISFI must examine the role-descriptors for **necessary** and **sufficient conditions.** These are predicate nodes in the domain knowledge networks that indicate when an object should or should not fill a role of another object. These nodes are also treated such that when toggled TRUE and FALSE they trigger potential consequences of the original mutation. An area of network around the necessary and sufficient conditions is copied into the problem network and the laws of the copied constraints are examined for potential consequences. Of course, these consequences are also mutations and may have further consequences.

Time Ordering.

The ability to order side effects temporally in a program is an absolute necessity and one that ISFI handles elegantly. In ISFI, each network has a hierarchy of states in which live all the nodes of the network. Transitions from state to state declare the gross flow of control for the program while at the same time rendering a partial temporal ordering of the mutations in those states. Confining mutations to states is another way of limiting the consequences of each side effect. A mutation (and its consequences) is performed in a state with the assurance that upon exiting the state (transitioning to another) all necessary actions for that mutation have been performed. An example state diagram is shown below. Note that ISFI allows users to discuss control flow of a program in this way without worrying about irrelevant details.



State Diagram

Control flow is determined by transitions
Scoping is determined by hierarchy links

Planning and Assumptions.

Planning and assumptions in ISFI are handled by the same mechanism that provides overall control and problem solving capabilities for the system. This mechanism plans, both according to stereotypical scenarios and according to its own problem solving skills, the coordinated use of ISFI's abilities, the execution of its own plans (via meta-plans), and attempts to recover from errors. The use of ISFI's abilities is demand driven. That is, no mechanism will be utilized until needed. The planning mechanism is best exemplified in controlling combinatorial explosion in applying transformations by matching the transformations' goal descriptors with current needs.

The primary form of assumption that ISFI uses is that some node has a certain value. For example, to avoid divide-by-zero errors ISFI checks the denominator of a division and makes sure it is not zero. If this fact can not be determined, then ISFI assumes (subject to more pressing demands) that the number is not zero. ISFI is willing to explain the assumptions that were made as well as to backtrack if some assumption proves false or unsuitable.

Code Generation and Program Building Blocks.

After all is said and done, the purpose of an automatic programming system is to produce programs written in some desired language(s). To this end, ISFI explicitly represents program building blocks to organize and store the program being synthesized. In other words, when ISFI needs to deal with a program or iteration, for example, ISFI manipulates the corresponding building block. In this way, one can view a program while it is being written. At various stages, ISFI selects, assembles, and fills with values these program building blocks. When completed, the building blocks are used in the generation of actual code. For this reason, ISFI actually creates a program internally and the code is generated from this internal representation; by the time code is produced, all the hard work is over. The building blocks can also be altered to represent code-level transformations. See [Standish76] for a catalog of source-to-source transformations that represent the flavor of transformations at this level (manipulating conditional expressions, moving loop invariants, boolean transformations).

ISFI is designed throughout to be target language independent. The program that is assembled in the building blocks remains the same regardless of what language the code will be written in. This property of ISFI is achieved by an overall model of computation that is itself free from any target language. The part of ISFI that is actually responsible for producing code is quite small, modular, and easily changed. For example, each building block knows of some way to produce the code for the part of the program that it contains. Changing target languages at this level is as simple as changing one function for each type of building block (currently there are about seven).

Example.

Recall the square root specification network above. The behavior that we desire from ISFI in writing the square root program should proceed as follows, guided by the agendas mechanism:

1. Attempt to find an algorithmic computation path in the network via propagation. Propagation will fail in the current example since the addition constraint has no law that will compute the value of B from the value of A.
2. Adapt the specification so that a computation path does exist. In this example, the shortest set of transformations is paraphrased as discovering that the value of B is bounded in both the high and low directions, defining an interval for B.
3. Based on the goal of finding the value of B, introduce an algorithm for repeatedly partitiioning the interval so that the value of B can be further limited on each iteration until B is known to within some assumed error bound. Assumptions are managed by agendas and used to fill in missing specification.
4. Code the algorithm in the desired target language allowing for syntax, variable semantics, local optimizations, etc.

This example provides an overview of how ISFI proceeds through a programming problem. Keep in mind that this example did not make use of data structures, side effects, subroutines, or any complex control flow when, in fact, ISFI codes examples that demonstrate the use of these types of programming knowledge.

Conclusion.

ISFI's goals for the first three years were primarily to extend the work done in [Brown81] on propagation and transformation to non-numeric domains and to address side effects. In these areas, ISFI has scored a success, coding programs in non-numeric domains such as a map display for a color monitor and maintaining data consistency in the presence of non-monotonic behavior ([Brown85B]). In the further areas of problem solving, automating algorithm and data structure design, and addressing realistically sized problems it's too soon to tell. Preliminary results are quite promising. ISFI has written programs of moderate size and varying complexity in a number of domains. This paper attempts to describe ISFI's mechanisms, why they are important, and why their implementations and interactions are special.

Bibliography

[Aho79]
Alfred V. Aho, Jeffrey D. Ullman
Principles of Compiler Design
Addison-Wesley Publishing Company, Reading, MA
1979

[Balzer83]
R. Balzer
Specification-Based Computing Environments
Information Sciences Institute memo, March 1983

[Barstow83]
D. Barstow
A Perspective on Automatic Programming
IJCAf-83.

[Brachman83]
R. Brachman, R. Fikes, H. Levesque
Krypton: A Functional Approach to Knowledge Representation
IEEE Computer, Vol. 16, No. 10, Oct. 1983

[Brown81]
R. Brown
Coherent Behavior from Incoherent Knowledge Sources in the Automatic Synthesis of Numerical Computer Programs
MIT-AI-TR-610, January 1981

[Brown85A]
R. Brown
Automation of Programming: the ISFI Experiments
MITRE Corp. Document M85-21, June 1985

[Brown85B]
R. Brown, G. Cleveland
Mutations and their Consequences; a Study of Non-monotonic Behavior
MITRE Corp. Document M85-18, June 1985

[Boyle84]
J. Boyle, M. Muraldharan
Program Reusability through Program Transformation
IEEE Trans. on Software Engineering, Vol. SE10, No. 5, September 1984

[Davis80]
R. Davis
Meta-Rules: Reasoning about Control
AIJ 15, p. 179-222, Dec. 1980

[Doyle78]
J. Doyle
Truth Maintenance Systems for Problem Solving
MIT-AI-TR-419, January 1978

[Doyle79]
J. Doyle
A Glimpse of Truth Maintenance
in Artificial Intelligence: An MIT Prespective, Winston and Brown (eds), MIT press, 1979

[Frenkel85]
Karen A. Frenkel
Toward automating the software-development cycle.
CACM Vol. 28, No. 6, June 1985

[FRL77]
R. Bruce Roberts, Ira P. Goldstein
The FRL Manual
Memo 409, Artificial Intelligence Lab, MIT, 1977

[Green81A]
C. Green, J. Phillips, S. Westfold, T. Pressburger, B. Kedzierski, S. Angebranndt, B. Mont-Reynaud, S. Tappel
Research on Knowledge-based Programming and Algorithm Design -- 1981
KES-U-81-2

[Green81B]
C. Green, S. Westford
Knowledge-based Programming Self Applied
in Machine Intelligence 10, John Wiley & Sons, New York, 1982

[Kant83]
E. Kant
On the Efficient Synthesis of Efficient Programs
AIJ Vol. 20, p. 253-305, 1983

[Martin79]
W. Martin
Descriptions and the Specialization of Concepts
in Artificial Intelligence: An MIT Prespective, Winston and Brown (eds), MIT press, 1979

[McDermott80]
D. McDermott, J. Doyle
Non-Monotonic Logic I
AIJ, 13, p. 41-72, April 1980

[McDermott83]
D. McDermott
Contexts and Data Dependencies: A Synthesis
IEEE Trans. on PAMI, Vol. PAMI-5, No. 3, May 1983

[Minsky75]
M. Minsky
A Framework for Representing Knowledge
in Psychology of Computer Vision, P. Winston (ed) McGraw-Hill, 1975.

[Rich81]
C. Rich
Inspection Methods in Programming
MIT-AI-TR-604, June 1981

[Steele78]
Guy L. Steele
Rabbit: A Compiler for SCHEME
AI-TR-474, 1978
Artificial Intelligence Laboratory, MIT

[Swartout83]
W. Swartout
XPLAIN: a System for Creating and Explaining Expert Consulting Programs
AIJ, 21, p. 285-325, 1983

UN SYSTEME FORMEL DE CARACTERISATION DE L'EVOLUTION DES CONNAISSANCES

CHOURAQUI Eugène

Centre National de la Recherche Scientifique
Groupe Représentation et Traitement des Connaissances
31, Chemin Joseph Aiguier - 13402 Marseille Cédex 9 - France

RESUME.

Nous présentons dans cette communication un système formel permettant de caractériser l'évolution des connaissances. Pour ce faire nous introduisons en plus des connecteurs de la logique classique quatre connecteurs temporels - le futur immédiat, le futur médiat, le passé immédiat et le passé médiat - exprimant les transformations que peuvent subir les données dans le temps. L'axiomatisation de ces connecteurs et leur caractérisation sémantique nous ont conduits à définir un modèle d'interprétation de ce système formel comparable à celui que définit Kripke pour les logiques modales. Ce modèle d'interprétation a permis de prouver la consistance intrinsèque et la validité de ce système formel. Et nous avons élaboré une procédure de décision fondée en particulier sur l'axiomatisation des connecteurs temporels.
Ce système formel est un des modules qui compose le meta-système expert ARCHES.Il permet de représenter et de traiter le temps dans la base de connaissances associée.

ABSTRACT.

In this paper we present a formal system in order to characterise the evolution of knowledge. In addition to the connectives of classical logic, we introduce four temporal connectives - the mediate future, the immediate future, the mediate past and the immediat past - expressing the transformations that may affect data in the course of time. The axiomatisation of these connectives and their semantic characterisation lead us to define a model of interpretation for the formal system which is comparable to that of Kripke for modal logic. With this model we prove the intrinsic consistence and the validity of this formal system. Similarly we have elaborated a decision procedure particularly based upon the temporal connectives.
This formal system is one of the component modules of the meta-expert system ARCHES. It allows the representation and the processing of the time in the corresponding knowledge base.

DOMAINES : Représentation de connaissances, Raisonnement formel.

INTRODUCTION.

Cette communication concerne la description formelle et les propriétés logiques d'un système formel particulier - noté S_Δ - permettant d'exprimer les modalités d'évolution des connaissances. Ce système formel constitue un des modules qui compose le système ARCHES, système symbolique de représentation et de traitement de connaissances, [1]-[3]-[4].
Pour représenter dans ARCHES l'évolution des faits étudiés, nous avons été conduits à développer une logique non classique dans laquelle ont été définis des connecteurs temporels,[5]-[9].Chaque fait est représenté par une formule donnant une description d'un individu ; ainsi si A est un symbole de concept, x une variable ou une constante d'individu et y une description, alors A(x,y) est une formule du système symbolique ARCHES, [2].
Plus précisément ces descriptions déterminent un système formel S_Δ composé de quatre éléments qui expriment les deux aspects "Représentation" et "Déduction" respectivement les définissant et les organisant :

$$S_\Delta = \{\mathcal{A}, \Delta, \Delta_T, ==>\}$$

$\mathcal{A}$ est un alphabet à partir duquel est engendré l'ensemble Δ des descriptions ; $\Delta_T \supset \Delta$ représentant l'ensemble des thèses de S_Δ. Ces trois ensembles, qui ont pour objet de définir et de véhiculer la représentation des descriptions, sont construits à partir d'une structure de données primitive permettant de véhiculer les descriptions élémentaires appelées termes descriptifs. La relation ==> est une relation de déduction qui détermine l'activité inférentielle qui peut être développée sur les descriptions. Elle est déterminée d'une part à partir de relations de réécriture définies sur les termes descriptifs et d'autre part à partir de conditions spécifiques qui la relient à deux catégories de connecteurs : d'une part les connecteurs définissant l'addition, la disjonction et la négation et d'autre part les connecteurs temporels définissant le futur immédiat, le futur médiat, le passé immédiat et le passé médiat. Nous donnons une caractérisation sémantique de ce système formel temporel en définissant un modèle d'évolution comparable à celui défini par Kripke pour la logique modale, [6]-[7]-[8] ; et nous montrons que la relation de déduction ==> est valide par rapport à cette interprétation. Enfin nous présentons sommairement une procédure de décision pour la relation ==>.

SYSTEME DE REPRESENTATION DES DESCRIPTIONS.

Les éléments primitifs.

Les termes descriptifs permettent de représenter les propriétés et d'une manière plus générale les relations qui caractérisent les individus. Ils sont construits à partir de quatre entités basiques : le trait, la classe, l'opérateur et le symbole fonctionnel $. Les traits permettent de représenter les caractères distinctifs des individus.Ceux de même nature sémantique sont regroupés dans des ensembles nommés classes. Les symboles de classes expriment la portée sémantique des propriétés ou des relations qui sont attestées dans les descriptions élémentaires.Les rapports qui existent entre les classes (considérées comme des pseudo-variables) et les traits sont exprimés par des symboles fonctionnels particuliers, en général n-aires, appelés opérateurs. Par exemple, l'énoncé "Automobile dont la capacité est inférieure ou égale à 5" montre le type de rapport qui peut exister entre classes et traits : le trait "nombre de places" (i.e. valeur numérique 5) est relié à sa classe Capacité par l'opérateur arithmétique .LE.(≤). Par ailleurs les propriétés et les relations peuvent être décrites localement comme le montrent les énoncés suivant : "Amphores ayant un timbre T_1 de type P_1" et "Automobile ayant une couleur bleu-foncé". La caractérisation locale des traits par des termes descriptifs permet précisément de représenter cette situation descriptive (propriétés de propriétés, relations précisées par des propriétés, relations de relations, etc.). Le lien qui exprime localement l'attribution d'un élément de description à un trait est représenté par le symbole fonctionnel $. Quand une propriété ou une relation n'est pas décrite localement, on dira que le trait correspondant est caractérisé par la description vide.

Un terme descriptif est une expression de la forme $op_n(T,t_1)$ dans laquelle : op_n est un opérateur n-aire, T un symbole de classe, et enfin t_1 un tuple de degré n dont les éléments sont soit de la forme (t_i, λ), soit de la forme $(t_i,op_{ni}(T_i,t_{1i}))$ où t_i est un trait faisant référence à la classe T, λ la description vide et $op_{ni}(T_i,t_{1i})$ un terme descriptif qui caractérise localement t_i.

Si les éléments du tuple t_i sont de la forme (t_i, λ), on dira que le terme descriptif est un terme descriptif non décrit ; dans le cas contraire on dira qu'il s'agit d'un terme descriptif décrit. Ainsi dans l'énoncé "L'automobile V de couleur bleu-foncé est dans le garage G", l'individu V, instance du concept Automobile, est caractérisé par les deux termes descriptifs Isa(Couleur,$(Bleu,Isa(Teinte,Foncé))) et In(Localisation,Ins(Garage,G)), le premier étant décrit et le second non décrit.

L'ensemble Δ des descriptions.

On appelle description d'un individu x appartenant à l'extension du concept C l'ensemble organisé des termes descriptifs qui le caractérise.

Plus précisément les descriptions sont engendrées à partir d'un alphabet $\mathcal{A}$ formé de quatre catégories de symboles : /1/ un ensemble dénombrable $\mathcal{L}$ de termes descriptifs : /2/ la chaîne vide λ ; /3/ deux catégories de connecteurs : d'une part les connecteurs classiques ¬, ∧ et ∨ nommés respectivement Négation, ET d'addition et OU non exclusif, et d'autre part les connecteurs temporels +, $\oplus$, - et $\ominus$ nommés respectivement futur immédiat, futur médiat, passé immédiat et passé médiat ; /4/ enfin les parenthèses (et).

$$\mathcal{A} = \mathcal{L} \cup \{\lambda\} \cup \{\neg, \wedge, \vee, +, \oplus, -, \ominus\} \cup \{(,)\}$$

L'ensemble dénombrable Δ des descriptions est un langage formel construit sur $\mathcal{A}$. Les descriptions, qui en sont les formules bien formées, obéissent à des règles de formation dont la définition est tout à fait classique.

Les thèses du système formel S_Δ.

Soit $\mathcal{J}$ l'ensemble fini des individus et $\mathcal{D}$ l'application qui permet d'associer à tout x sa description $\mathcal{D}(x)$. Si l'individu x est une instance du concept C, nous avons vu que l'attribution $\mathcal{D}(x)$ à x est représentée par la structure $C(x, \mathcal{D}(x))$: nous dirons que $\mathcal{D}(x)$ est une thèse du système formel S_Δ.Soit $\Delta_T \supset \Delta$ l'ensemble des descriptions tel que l'application $\mathcal{D}$ soit une bijection de $\mathcal{J}$ sur Δ_T : les structures stockées dans la base de connaissances du système ARCHES sont les formules toujours vraies du type $C(x, \mathcal{D}(x))$ dans lesquelles x instance du concept C a pour description $\mathcal{D}(x) \in \Delta_T$.

Δ_T est l'ensemble des thèses du système formel S_Δ de caractérisation des descriptions.

ORGANISATION DEDUCTIVE DES DESCRIPTIONS.

Les règles de transformation des termes descriptifs.

La représentation et les propriétés logiques des termes descriptifs permettent de définir sur ces derniers trois types de règles de réécriture : les règles de décomposition, les règles d'héritage et de transitivité, et enfin les règles d'extension. Ces règles expriment les propriétés sémantiques des classes. Par exemple les rapports sémantiques qui existent entre les classes Partie et Matériau sont exprimés par la règle d'héritage ci-après :

Isa(Partie,$(x,Isa(Matériau,y))) -->
Isa(Matériau,y)

Cette règle indique que si un individu possède une partie x dont le matériau est y, alors le matériau avec lequel est fabriqué cet individu est également y.

D'une manière générale, on se donne pour toute application plusieurs règles de réécriture du type précédent. Ces règles définissent le système de règles de substitution-réduction du système ARCHES, à partir duquel sont produites les transformations des termes descriptifs. Ces transformations sont déterminées par la relation de réécriture -->* définie sur l'ensemble des termes descriptifs comme la fermeture transitive et réflexive de la relation -->.

Nous avons démontré que les résultats des opérations de dérivation sont indépendants de l'ordre d'application des règles de réécriture. Ceci nous a permis de construire un algorithme original de décidabilité pour la relation A-->*B.

Définition de la relation ==>.

Pour compléter la définition du système formel S_Δ, nous devons définir les modalités de dérivation des descriptions, compte tenu d'une part de leurs

règles de formation à partir des termes descriptifs et des connecteurs $\wedge$, $\vee$, $\neg$, +, $\oplus$, - et $\ominus$, et d'autre part des relations de réécriture des termes descriptifs. Pour ce faire nous définissons sur l'ensemble Δ une relation de déduction entre les descriptions, notée ==>, qui est la plus petite relation réflexive et transitive vérifiant non seulement les conditions classiques relatives aux connecteurs de la logique du premier ordre ($\wedge$, $\vee$, $\neg$), mais aussi les conditions suivantes, non classiques, liées à la définition des connecteurs temporels (voir par exemple sur ce sujet [10]) :

C1 $a \Rightarrow \oplus a$
C1' $a \Rightarrow \ominus a$
C2 $+a \Rightarrow \oplus a$
C2' $-a \Rightarrow \ominus a$
C3 $+ \oplus a \Rightarrow \oplus +a$
C3' $- \ominus a \Rightarrow \ominus - a$
C4 $\oplus (a \vee b) \Rightarrow \oplus a \vee \oplus b$
C4' $\ominus (a \vee b) \Rightarrow \ominus a \vee \ominus b$
C5 $\oplus \neg \ominus \neg a \Rightarrow a$
C5' $\ominus \neg \oplus \neg a \Rightarrow a$

Il est à noter en particulier que les conditions C1, C2, C1' et C2' expriment d'une part que le présent et le futur immédiat font partie du futur et d'autre part que le présent et le passé immédiat font partie du passé.

La relation d'égalité =.

On dira que a=b si et seulement si a ==> b et b ==> a.La relation =, et donc également la relation ==>, satisfait aux conditions classiques des connecteurs $\wedge$, $\vee$ et $\neg$, et aux conditions suivantes liées à la définition des connecteurs temporels :

C6 $+ \neg a = \neg + a$
C6' $- \neg a = \neg - a$
C7 $+ (a \wedge b) = + a \wedge + b$
C7' $- (a \wedge b) = - a \wedge - b$
C8 $\oplus \oplus a = \oplus a$
C8' $\ominus \ominus a = \ominus a$
C9 $\oplus \lambda = \lambda$
C9 $\ominus \lambda = \lambda$
C10 si a ==> b alors +a ==> +b
C10' si a ==> b alors - a ==> - b
C11 si a ==> b alors $\oplus$ a ==> $\oplus$ b
C11' si a ==> b alors $\ominus$ a ==> $\ominus$ b
C12 $- + a = + - a$
C13 $\ominus \oplus a = \oplus \ominus a$

Il est à noter tout d'abord que le quotient $\Delta/=$ est un treillis distributif complémenté du fait des propriétés sémantiques des connecteurs $\wedge$, $\vee$ et $\neg$.
Par ailleurs les conditions C6 et C6' expriment que l'évolution des descriptions est déterministe (les quantifications universelle et existentielle sont confondues, i.e. $\neg + \neg = +$ et $\neg - \neg = -$). Les conditions C8 et C8' indiquent que les connecteurs $\oplus$ et $\ominus$ sont transitifs. Par ailleurs C9 et C9' expriment la cohérence de l'évolution des descriptions, en ce sens que l'ensemble Δ possède un minimum et un seul. Les conditions C10, C11, C10' et C11' indiquent que si la formule a ==> b est vérifiée alors elle est également vérifiée dans le futur ou dans le passé. Enfin les conditions C12 et C13 expriment la symétrie des connecteurs définissant les futurs et les passés.

CARACTERISATION SEMANTIQUE DU SYSTEME FORMEL S_Δ.

Définition.

On appelle interprétation du système formel $S_\Delta = \{\mathcal{A}, \Delta, \Delta_T, ==>\}$ la structure $\mathcal{H} = \{D, \mathcal{C}\}$ dont les éléments D et $\mathcal{C}$, représentant respectivement un ensemble et une fonction, sont définis ci-après.

Définition de D.

D est un ensemble non vide appelé domaine d'interprétation de S_Δ. L'ensemble D est formé de l'union de deux ensembles D_i et D_c : les éléments du premier correspondent aux symboles d'individus appartenant à l'ensemble $\mathcal{J}$, et les éléments du deuxième correspondent aux symboles de concepts. Ces deux ensembles possèdent les deux propriétés suivantes : d'une part $D_c \supset \mathcal{P}(D_i)$ ($\mathcal{P}(D_i)$ désigne l'ensemble des parties de D_i) car tout concept est interprété comme un ensemble d'individus ; d'autre part $D_c \supset D_i$ car tout concept peut être interprété comme un individu.

Définition de $\mathcal{C}$.

$\mathcal{C}$ est une fonction de correspondance qui permet d'associer à chaque composante du système formel S_Δ son interprétation dans le domaine D.

L'interprétation des individus, des concepts, des traits et du symbole fonctionnel $ ne dépendant pas du temps, nous ne donnons ci-après que celle qui caractérisent les modalités des faits caractérisant les individus.

(1) Interprétation des termes descriptifs formant la base $\mathcal{L}$ de Δ .
L'ensemble dénombrable Δ des descriptions est construit, en particulier, à partir de l'ensemble dénombrable $\mathcal{L}$ de termes descriptifs. Tout élément appartenant à $\mathcal{L}$ est de la forme $op_n(C, \#(t_1, \ldots t_i, \ldots t_n))$ dans laquelle op_n est un opérateur n-aire, C une classe, et enfin $\forall i \in [1,n]$ t_i est un trait décrit ou non décrit. Nous nous proposons de définir l'interprétation des éléments appartenant à $\mathcal{L}$ compte tenu des modalités d'évolution des individus que nous définissons ci-après.

Les modalités d'évolution des individus sont déterminées à partir de la définition d'un ensemble Ω isomorphe à l'ensemble N des entiers rationnels. Tout élément de Ω sera appelé état. Les états marquent les différentes transformations que peuvent subir les descriptions caractérisant les individus. Chaque état, qui exprime la situation descriptive des individus à un moment donné, sera repéré par un entier positif, négatif ou nul. Dans notre modèle d'interprétation nous supposons que l'évolution est déterministe car Ω est isomorphe à N (conditions C6 et C6') ; nous entendons par là le fait que chaque état possible ne peut évoluer que vers un état et un seul. Ainsi à partir d'un état initial E_o les interprétations des descriptions caractérisant les individus dans l'état E_o évoluent de telle manière que les

transformations qu'elles subissent engendrent une séquence linéaire d'états successifs organisés par la structure $\mathcal{K}$, organisation qui est de même nature que celle définie par Kripke pour les logiques modales, [8].
L'interprétation d'un terme descriptif appartenant à $\mathcal{L}$ est définie comme l'ensemble des interprétations des individus qui sont caractérisés par ce terme descriptif ; et cette interprétation tient compte de l'évolution de ces individus qui sera exprimée à l'aide de l'ensemble N conformément à la définition de Ω.
En d'autres termes, l'interprétation de tout couple <classe,opérateur> de la forme $<C,op_n>$ détermine une application de T^m dans $(N\text{-------}>\mathcal{P}(D_i))$:

$(<C,op_n>) : \quad T^m \text{-----}> (N \text{ -----}> \mathcal{P}(D_i))$

avec $\mathcal{C}(op_n(C,\#(t_1,\ldots,t_n)))(i) = \mathcal{C}(<c,op_n>)(\mathcal{C}(t_1),\ldots,\mathcal{C}(t_n))(i) \in \mathcal{P}(D_i)$

L'union des ensembles de la forme $\mathcal{C}(op_n(C,\#(t_1,\ldots,t_n)))(i)$ pour toutes les interprétations des couples <classe,opérateur> forme le ième état. En particulier si pour l'interprétation du couple $<C,op_n>$ on a $\mathcal{C}(op_n(C,\#(t_1,\ldots,t_n)))(i) = \phi$ alors aucun individu n'est caractérisé par le terme descriptif $op_n(C,\#(t_1,\ldots,t_n))$ dans l'état i.

(2) Interprétation des descriptions.
Toute description a est interprétée comme l'ensemble des interprétations des individus qui sont caractérisés par a : plus précisément, elle est interprétée comme une application de N dans $\mathcal{P}(D_i)$ pour tenir compte de l'évolution des individus :

$$\Delta \text{ --------}> (N \text{ --------}> \mathcal{P}(D_i))$$

avec

$$\mathcal{C}(a)(i) \in \mathcal{P}(D_i)$$

$\mathcal{C}(a)(i)$ détermine l'ensemble des interprétations des individus qui sont caractérisés par la description a dans l'état i.
Nous désignerons par $\mathcal{C}_o$ l'application qui détermine l'interprétation des descriptions à l'état initial E_o :

$$\mathcal{C}_o(a) = \mathcal{C}(a)(o)$$

Les règles de formation des descriptions à partir des connecteurs $\wedge$, $\vee$, $\neg$, $+$, $\oplus$, $-$ et $\ominus$ ainsi que la définition de $\mathcal{C}_o$ contribuent à déterminer par récurrence la fonction de correspondance $\mathcal{C}$ à l'aide des règles ci-après :

(i) *Interprétation des connecteurs classiques.*

1 $\mathcal{C}(a \wedge b)(n) = \mathcal{C}(a)(n) \cap \mathcal{C}(b)(n)$
2 $\mathcal{C}(a \vee b)(n) = \mathcal{C}(a)(n) \cup \mathcal{C}(b)(n)$
3 $\mathcal{C}(\neg a)(n) = \begin{cases} \mathcal{C}(a)(n) \\ \text{complément dans } D_i \end{cases}$ ($\begin{cases} \\ \end{cases}$:

(ii) *Interprétation des connecteurs temporels.*
4 $\mathcal{C}(+a)(n) = \mathcal{C}(a)(n+1)$
5 $\mathcal{C}(\oplus a)(n) = \bigcup_{p=o}^{+\infty} \mathcal{C}(a)(n+p)$
6 $\mathcal{C}(-a)(n) = \mathcal{C}(a)(n-1)$
7 $\mathcal{C}(\ominus a)(n) = \bigcup_{p=o}^{-\infty} \mathcal{C}(a)(n+p)$
8 $\mathcal{C}(\lambda)(n) = \phi$

Les trois premières règles correspondent à l'interprétation classique et intuitive des connecteurs $\wedge$, $\vee$ et $\neg$.

Les règles 4,5,6 et 7 déterminent la sémantique de l'évolution des descriptions : l'interprétation des connecteurs + et - montre que ces derniers permettent d'exprimer l'évolution entre deux états consécutifs E_i et E_{i+1} ou E_i et E_{i-1} ; celle des connecteurs $\oplus$ et $\ominus$ montre que ces derniers expriment l'évolution entre deux états successifs E_i et E_j ($j \geq i$ ou $j \leq i$), c'est-à-dire non nécessairement consécutifs. De même les règles 5 et 7 permettent de déduire que $\mathcal{C}(\oplus \oplus a)(n) = \mathcal{C}(\oplus a)(n)$ et $\mathcal{C}(\ominus \ominus a)(n) = \mathcal{C}(\ominus a)(n)$, ce qui prouve la transitivité des connecteurs $\oplus$ et $\ominus$. Enfin remarquons que, contrairement aux connecteurs + et -, les connecteurs $\oplus$ et $\ominus$ supposent que le présent fait partie du futur et du passé (car $j \geq i$ ou $j \leq i$). Par ailleurs la règle 8 indique que, quelque soit l'état E_i, il n'existe aucun individu x qui soit caractérisé par la description vide.

L'introduction des connecteurs $\oplus$ et $\ominus$ est indispensable pour la dérivation des descriptions dans lesquelles la séquence des états n'est pas explicitée (on recherche au moins l'existence d'un état futur ou passé dans lequel un individu x a tel ou tel groupe de propriétés ; ces connecteurs sont analogues à la quantification existentielle de la logique classique).

VALIDITE DE LA RELATION DE DEDUCTION ==>.

Définition 1.
Etant donné deux descriptions quelconques a et b appartenant à Δ, on dira que la formule a ==> b est valide si pour toute interprétation $\mathcal{M}$ et pour tout i $\mathcal{C}(a)(i)$ est inclus dans $\mathcal{C}(b)(i)$:

$$\mathcal{C}(a)(i) \supset \mathcal{C}(b)(i)$$

Définition 2.
On dira que la relation de déduction ==> est valide si toutes les formules qui la composent sont valides.

Théorème.

La relation de déduction ==> est valide.

Démonstration. Pour démontrer que la relation de déduction ==> est valide il suffit de démontrer que toutes les conditions auxquelles cette relation satisfait sont valides.
La validité des conditions relatives aux connecteurs classiques est immédiatement établie car le quotient $\Delta/=$ est un treillis distributif complémenté.
Prouvons la validité des conditions C1 à C11, C1' à C11' et enfin C12 et C13.
Preuve de C1. $\mathcal{C}(\oplus a)(i) = \bigcup_{p=o}^{\infty} \mathcal{C}(a)(i+p)$ d'après

la règle 5 ; d'où $\mathcal{C}(a)(i) \supset \mathcal{C}(\oplus a)(i)$, ce qui prouve la validité de la formule $a \Longrightarrow \oplus a$.

Preuve de C2. $\mathcal{C}(\oplus a)(i) = \bigcup_{p=o}^{\infty} \mathcal{C}(a)(i+p)$; or $\mathcal{C}(+a)(i) = \mathcal{C}(a)(i+1)$. Il en résulte que $\mathcal{C}(+a)(i) \supset \mathcal{C}(\oplus a)(i)$, ce qui prouve la formule $+a \Longrightarrow \oplus a$.

Preuve de C3. On a simultanément d'une part

$\mathcal{C}(\oplus + a)(i) = \bigcup_{p=o}^{\infty} \mathcal{C}(+a)(i+p) = \bigcup_{p=o}^{\infty} \mathcal{C}(a)(i+1+p)$, et d'autre part $\mathcal{C}(+ \oplus a)(i) = \mathcal{C}(\oplus a)(i+1) = \bigcup_{p=o}^{\infty} \mathcal{C}(a)(i+1+p)$. Il en résulte en particulier que la formule $+\oplus a \Longrightarrow \oplus +a$ est vérifiée.

Preuve de C4. On a simultanément d'une part : $\mathcal{C}(\oplus (a \vee b))(i) = \bigcup_{p=o}^{\infty} \mathcal{C}(a \vee b)(i+p)$, et d'autre part : $\mathcal{C}(\oplus a \vee \oplus b)(i) = \mathcal{C}(\oplus a)(i) \cup \mathcal{C}(\oplus b)(i) = \bigcup_{p=o}^{\infty} (\mathcal{C}(a)(i+p) \cup \mathcal{C}(b)(i+p)) = \bigcup_{p=o}^{\infty} \mathcal{C}(a \vee b)(i+p)$. Il en résulte en particulier que la formule $\oplus (a \vee b) \Longrightarrow \oplus a \vee \oplus b$ est vérifiée.

Preuve de C5. La formule $\oplus \neg \ominus \neg a \Longrightarrow a$ est valide car $\mathcal{C}(a)(n) = \mathcal{C}(a)(n)$.

Preuve de C6. Pour prouver la validité de la formule $+\neg a = \neg +a$ il suffit de montrer que $\forall i \quad \mathcal{C}(\neg +a)(i) = \mathcal{C}(+\neg a)(i)$. Nous avons d'après la règle 4 $\mathcal{C}(+\neg a)(i) = \mathcal{C}(\neg a)(i+1)$; d'où $\mathcal{C}(\neg a)(i+1) = \mathcal{C}(a)(i+1)$ d'après la règle 3. Il en résulte que $\mathcal{C}(+\neg a)(i) = \mathcal{C}(+a)(i)$ d'après la règle 4, d'où : $\mathcal{C}(+\neg a)(i) = \mathcal{C}(\neg +a)(i)$ soit $\mathcal{C}(+\neg a)(i) = \mathcal{C}(\neg +a)(i)$ (c.q.f.d.).

Preuve de C7. $\mathcal{C}(+(a \wedge b))(i) = \mathcal{C}(a \wedge b)(i+1)$ d'après la règle 4 ; d'où $\mathcal{C}(+(a \wedge b))(i) = \mathcal{C}(a)(i+1) \cap \mathcal{C}(b)(i+1)$ d'après la règle 1. Il en résulte que $\mathcal{C}(+(a \wedge b))(i) = \mathcal{C}(+a)(i) \cap \mathcal{C}(+b)(i)$, soit $\mathcal{C}(+(a \wedge b))(i) = \mathcal{C}(+a \wedge +b)(i)$ (c.q.f.d.).

Preuve de C8. Evident.

Preuve de C9. Evident car $\forall i$ on a $\mathcal{C}(\lambda)(i) = \phi$.

Preuve de C10 et C11. Evident d'après la définition 1 et les règles 4 et 5.

Le même procédé de démonstration permet de démontrer les conditions C1' à 11', ainsi que les conditions C12 et C13.

Il en résulte que toutes les conditions auxquelles satisfait la relation ==> sont valides. On en déduit d'après la définition 2 que la relation ==> est valide (c.q.f.d.).

METHODE DE RESOLUTION DE LA FORMULE H ==> C.

Dans le but de représenter les descriptions sous une forme canonique exprimée comme une addition de disjonctions, nous avons été amenés à faire l'hypothèse que les descriptions du type $\oplus (a \wedge b)$, $\oplus \neg(a \vee b)$, $\ominus (a \wedge b)$ et $\ominus \neg (a \vee b)$ ne sont pas des formules du système formel S_Δ.

Cette restriction nous a permis d'élaborer une procédure de décision permettant de résoudre le problème suivant : étant donné un couple de descriptions (H,C), déterminer s'il vérifie la relation H ==> C. Ce problème est évidemment essentiel pour la démonstration des théorèmes du système ARCHES. La définition de cette procédure, qui s'appuie sur les propriétés formelles de la relation ==>, utilise la méthodologie de résolution des problèmes par décomposition et construction de graphes "et/ou" correspondants. Plus précisément, cette procédure construit deux arbres "et/ou" $\mathcal{A}_h$ et $\mathcal{A}_c$ associés respectivement à l'hypothèse H et à la conclusion C, les modalités de construction étant déterminées à partir des schémas de dérivation des descriptions et de leurs propriétes. Elle construit ensuite l'arbre "et/ou" $\mathcal{A}_r$ en accrochant à chaque terminal de $\mathcal{A}_h$ l'arbre $\mathcal{A}_c$ sans sa racine ; et tente de valider la relation H ==> C en cherchant à valider au moins un sous-arbre "et" de $\mathcal{A}_r$ en utilisant en particulier l'algorithme de décidabilité de la relation -->*.

CONCLUSION.

La conception du système formel S_Δ dont les propriétés logiques ont été systématiquement étudiées est une contribution méthodologique et théorique à l'étude de l'évolution des connaissances. Nous nous sommes essentiellement intéressés à la représentation et au traitement du explicitement le temps dans la base de connaissances de ARCHES, mais aussi de déterminer le traitement formel de l'évolution de ces connaissances.

REFERENCES.

[1] BOURRELLY L. - Représentation et Simulation du raisonnement par système expert. Thèse de doctorat, Marseille, à paraître en 1986.

[2] CHOURAQUI E.- Contribution à l'étude théorique de la représentation des connaissances, le système symbolique ARCHES. Thèse de doctorat d'état, Nancy, Septembre 1981.

[3] CHOURAQUI E. - ARCHES, un système symbolique de représentation et de traitement de connaissances. Congrès AFCET Informatique 1981, Gif-sur-Yvette, 18-20 Novembre 1981.

[4] CHOURAQUI E. - Construction of a Model for Reasoning by Analogy. 1982 European Conference on Artificial Intelligence, Orsay, France, ECAI-82 Proceeding, 48-53, 12-14 Juin 1982.

[5] FARINAS DEL CERRO L. - Le temps et sa représentation. Colloque sur les processus de mémorisation, Aix-en-Provence, France, 26-28 Juin 1979.

[6] FARINAS DEL CERRO L. - Déduction automatique et logique modale. Thèse de doctorat d'état, Paris, Juin 1981.

[7] HUGHES, CRESWELL - An introduction to modal logic. Fletcher and Sons, Ltd Northwich, Great Britain, 1973.

[8] KRIPKE S. - Semantical considerations in modal logic. Acta Phylosophica Fennica, vol. 16, 83-94, 1963.

[9] MOORE R. - Reasoning about knowledge and Action. I.J.C.A.I. Proceedings, 223-227, 1977.

[10] PORTE J. - Recherches sur la théorie générale des systèmes formels. Gauthier-Villars, Paris, 1965.

Une expérience de l'Ingenierie de la Connaissance : CODIAPSY développé avec HAMEX

MAURY Michel* MASSOTTE A-M* BETAILLE Henri* PENOCHET J-C** NEGRE Michelle**

* C.R.I.M. 860 Rue de S^{t} Priest 34100 MONTPELLIER FRANCE
** G.R.I.P. Hôpital Colombière 34059 MONTPELLIER FRANCE

RESUME : Ce papier décrit une méthodologie pragmatique pouvant être utilisée par un ingénieur de la connaissance afin de conduire avec succès un projet de réalisation de système expert. Nous donnons un exemple illustrant cette approche et montrant son efficacité pour la création d'un système expert : CODIAPSY. Ce dernier est développé avec le moteur d'inférence HAMEX.

ABSTRACT : This paper is aimed at describing a methodology to be used in the knownledge engeneering in order to successfully design a knownledge base in an expert system. An example is detailed to illustrate how this approach was implemented in the CODIAPSY expert system and how they were efficient. HAMEX is the inference engine.

INTRODUCTION :

Notre travail sur les systèmes experts nous a conduit à définir, à concevoir et à implanter un moteur d'inférence, des interfaces, des logiciels d'aide pour le cogniticien/expert qui veut écrire une base de connaissances et, enfin, un certain nombre d'applications.

Dans cette présentation :
- la première partie décrit succintement HAMEX, notre moteur d'inférence (version juillet 85),
- la deuxième partie fait le point sur les procédures de contrôle qui ont été réalisées,
- la description de l'application CODIAPSY, en troisième partie, pour l'aide au diagnostic psychiatrique, appuie nos idées sur la conduite d'un projet et permet d'essayer de mettre au point et de perfectionner une certaine méthodologie.

I - LE MOTEUR D'INFERENCE HAMEX

Dans la version de juillet 85, le moteur d'inférence HAMEX présente les caractéristiques suivantes :
- deux versions sont disponibles : une version concepteur de systèmes experts et une version utilisateur.
- Il travaille sur une base de connaissances formée de règles de production.
- Il ne permet pas l'utilisation de "variables" mais autorise le test des valeurs des attributs, la comparaison de la valeur de deux attributs, la lecture de la valeur d'un attribut (valeur chaîne de 46 caractères au plus ou valeur réelle < 10^{38} avec 7 chiffres significatifs).
- Il fonctionne en chaînage avant avec recherche d'une ou plusieurs solutions et en chaînage arrière avec la possibilité d'utiliser uniquement les hypothèses.
- La base des connaissances est constituée d'une base de règles et d'une base de faits. Elle n'est pas figée car le concepteur peut y ajouter des ensembles de faits ou règles à tout moment. Il est possible d'avoir plusieurs bases de connaissances différentes.
- Les concepteurs de systèmes sont guidés dans leur tâche par l'affichage de menus hiérarchisés. Il n'est pas nécessaire de connaître un langage de commande pour dialoguer avec le moteur qui propose dans chaque cas la démarche à suivre.
- Il existe une trace concepteur qui permet une mise au point aisée en fournissant le nom des règles testées et appliquées, les modifications des valeurs des attributs ainsi que des conditions.
- Un système de pagination permet l'utilisation de toute la mémoire centrale disponible pour la base de connaissances, ce qui donne des temps de réponse très courts. La construction d'un arbre de recherche ordonnancé lors de l'introduction des règles y contribue aussi.
- Une connexion avec l'environnement informatique, y compris par ligne de communications, est possible par l'intermédiaire de fichiers de faits. Ces fichiers sont lus ou créés par l'exécution d'actions dans les règles.
- Il est écrit en langage PASCAL sous MS-DOS et fonctionne sur IBM-PC avec 256 K-octets de mémoire.

Les extensions en cours de tests permettront :
- l'utilisation de coefficients de vraissemblance pour les faits et dans chaque action d'une règle,
- la définition de plusieurs classes de règles et l'utilisation de métarègles lors des phases de filtrage et de choix,
- des expressions arithmétiques ou chaînes de caractères dans lesquelles toutes les principales fonctions du langage PASCAL seront utilisables.

Pour les variables à valeur chaîne de caractères, la valeur sera sur une longueur d'au plus 128 caractères.

- une gestion multi-fenêtres dans les actions des règles qui permettent des affichages sur l'écran ou la lecture d'informations au clavier,
- la gestion des règles, des hypothèses et des buts par un éditeur pleine page.

II - LES OUTILS D'AIDE A L'ECRITURE D'UNE BASE DE CONNAISSANCES

Remarque : Dans le suite de cet article, nous utilisons une série de "mots" dont les définitions sont les suivantes :

Donnée : désignera une information quelconque que l'homme transmet au système informatique pour qu'il la traite ou qu'il reçoit en réponse. Ce peut être aussi bien une règle, ou un terme, que la base de connaissances dans son ensemble.

Terme : non général désignant une action, une condition, une hypothèse ou une conclusion.

Faits : hypothèses du concepteur ou éléments de description du monde de l'utilisateur.

La qualité des conclusions obtenues par la résolution du problème posé dépend des données d'entrée, c'est-à-dire la base des connaissances. Cette dernière doit donc vérifier des contraintes d'intégrité.

La tâche la plus compliquée quand on développe une base de connaissances consiste à définir un ensemble de règles complet et cohérent. Prouver la cohérence est en général impossible.

Nous nous sommes donc attachés à fournir un environnement de conception de systèmes experts qui comprenne non seulement un mécanisme de résolution de problème, un interface de dialogue homme/machine mais encore un vérificateur de règles.

LES CONTROLES :

Dans le cas de la réalisation d'une base de connaissances, les contrôles ne peuvent être orientés uniquement sur les faits. Ils doivent surtout porter sur la surveillance des activités et sur la méthodologie suivie. Les différents contrôles effectués sont :

- le contrôle direct des données,
- les contrôles relatifs ou internes,
- le contrôle de l'état du système.

ANALYSE DES CONTROLES POSSIBLES AVEC LA REPRESENTATION DES CONNAISSANCES CHOISIE :

Du fait de l'utilisation de règles de production, il est facile au système d'expliquer son raisonnement, en parcourant simplement l'arbre de déduction qu'il vient de créer. Ceci permet de donner des explications à l'utilisateur mais cela permet aussi à l'expert de contrôler la qualité de la représentation de la connaissance.

On peut énumérer les cas de fautes qui paraissent suffisamment intéressants pour être relevés et qu'il est facile de mettre en évidence dans un système basé sur la logique :

- conflits :
  * compatibilité avec les règles déjà entrées,
  * vraissemblance par rapport à des seuils,
  * cohérence séquentielle avec la ou les règles précédentes : pour cela dans le cadre de HAMEX, il lui est offert trois possibilités de listes de règles ayant soit les mêmes prémisses, soit les mêmes conclusions, soit une condition d'arrêt. Ce dernier cas est important car les règles qui possèdent un terme STOP interrompent la déduction et il faut donc s'assurer qu'elles sont bien écrites.
- redondance,
- sous-ensembles,
- règles oubliées,
- complétude : en ce qui concerne la complétude, le logiciel met simplement en évidence l'existence d'un terme pendant et affiche la ou les règles dans lesquelles il se trouve.

Nous avons réalisé un logiciel mettant en évidence ces différentes lacunes que nous appelons le vérificateur. Quant il rencontre ces problèmes, il les assemble dans des tables que le concepteur de la base de connaissances analyse.

Dans ce qui précède nous n'avons fait mention que des problèmes que l'on peut rencontrer au cours de l'écriture des règles elles-mêmes. Mais de mauvaises conclusions fournies par le système peuvent découler des réponses données par l'utilisateur. On peut classer celles-ci en deux types possibles :

- soit l'utilisateur ne sait que répondre et donne une valeur fausse. Une vérification automatique de la validité de la réponse par rapport à des seuils et/ou le choix parmi une liste exhaustive de valeurs fournies par l'expert l'aiderait considérablement,
- soit le concepteur a oublié des cas, il faudra réviser la base de connaissances.

L'ECRITURE DES TERMES :

Il était nécessaire d'adjoindre au moteur un module d'interface qui permette d'assurer que le vocabulaire employé au niveau des règles avait un caractère d'unicité et de vérifier que celui qui était utilisé dans la description des faits était compatible. Le but poursuivi n'étant pas le traitement du langage naturel mais simplement le traitement des diverses formulations possibles d'un même terme afin de faciliter le transfert des informations. Un terme est constitué en général par une phrase courte dont la forme générale est la suivante :

<objet><prédicats><attribut><valeur>

Pour que le moteur puisse reconnaître comme égaux deux termes t1 et t2, il faut que ceux-ci soient rigoureusement identiques. La similitude est particulièrement difficile dans le cas où il s'agit d'une suite de caractères : un espace, une virgule, un "s" en plus ou mal placés suffisent à empêcher l'identification. Nous avons donc décidé d'adjoindre comme outil d'interface supplémentaire au moteur, un logiciel qui permet de faire abstraction :

- des espaces,
- de la ponctuation,

et qui met en évidence les termes identiques une fois cette opération effectuée. Il fait ressortir en plus les éléments qui ne diffèrent que d'un caractère (autre qu'un espace ou un signe de ponctuation).

LES ATTRIBUTS :

Dans le cadre d'une utilisation intensive des attributs à valeurs autres que booléennes, il a paru intéressant d'offrir la possibilité d'obtenir leur liste ordonnée sous forme de dictionnaire ainsi que les noms des règles dans lesquelles ils sont employés. Ce dictionnaire peut être fourni à la demande.

Par ailleurs, un contrôle des attributs, inséré directement dans HAMEX, vérifie qu'une valeur peut leur être attribuée par l'intermédiaire des règles ou des hypothèses. Ce contrôle est automatique, et précède chaque exécution qu'il refuse de faire s'il trouve un seul attribut non défini.

III - CODIAPSY, UNE EXPERIENCE D'INGENIERIE DE LA CONNAISSANCE

QU'EST-CE QUE LE DSM-III ?
(Diagnostic of Statistical Manual of Mental Disorders)

En raison de la complexité même de l'objet de la psychiatrie, la maladie mentale, la nosographie psychiatrique classique est :
- peu homogène,
- variable en fonction des conceptions théoriques des auteurs,
- les catégories diagnostiques diffèrent notablement d'un pays à l'autre.

Il faut ajouter que contrairement à la plupart des autres domaines médicaux, les critères objectifs qu'apportent les examens biologiques sont quasiment absents en psychiatrie.

Il fallait pourtant bien établir des systèmes de classification, des cadres communs de référence restants nécessaires notamment aux statistiques de santé publique nationale et internationale, en particulier à l'OMS (Organisation Mondiale de la Santé).

En France est utilisée, depuis plusieurs années une codification nationale qui est celle de l'ISERM et qui est compatible avec la codification internationale des maladies dans sa neuvième version (CIM 9).

Aux Etats-Unis, l'Association Américaine de Psychiatrie a proposé une nosographie unique, rédigée par un comité d'experts et périodiquement révisée : le DMS-III est la troisième édition, publiée en 1980, de cette classification des troubles mentaux.

Il présente une originalité à la fois sur le plan formel et sur le plan du contenu :
- sur le plan formel, il s'agit d'une classification multiaxiale (sur 5 axes),
- sur le plan du contenu, la terminologie diverge très souvent par rapport aux habitudes internationales. Le DSM-III contient un nombre important de classes nouvelles. Il est, par ailleurs, dénué de toute référence étiopathogénique.

Les critères de diagnostic ont pour buts essentiels de guider le clinicien dans l'utilisation d'une classification et dans la formulation d'un diagnostic. Ils facilitent la communication entre clinicien et permettent d'améliorer la méthodologie de la recherche psychiatrique. Mais ce procédé comporte certains inconvénients tels que :
- l'écueil d'une schématisation abusive,
- le risque d'une multiplication désordonnée de systèmes de classification,
- le fait d'imposer l'utilisation de critères non validés ou reposant sur des bases théoriques non partagées par l'ensemble de la communauté.

Le DSM-III était fourni sous la forme d'un modèle constitué par une série d'arbres de décision. Réunis, ils formaient un réseau du fait de l'interdépendance crée par l'existence de noeuds communs. Ceux-ci se traduisent, au niveau des résultats, par la notion de diagnostics multiples. L'ordonnancement des règles par le concepteur permet de représenter le classement des critères diagnostiques en principal et secondaire par rapport au malade considéré.

La traduction sous forme de propositions à valeurs VRAIE ou FAUSSE et ensuite leur insertion dans le modèle des règles de production de HAMEX n'a pas posé de gros problèmes, si ce n'est justement au niveau des éléments communs à plusieurs arbres. La facilité fournie par HAMEX de rechercher plusieurs solutions a été ici largement exploitée.

La mise au point du système a confirmé la nécessité et l'utilité de l'existence d'un certain nombre d'outils logiciels pour aider à augmenter la cohérence et la complétude des règles au moment de leur conception.

DEVELOPPEMENTS FUTURS :

Le système a été écrit essentiellement en utilisant la déduction, il serait intéressant (en particulier au niveau d'un médecin consultant ou au niveau de la formation) de raisonner en chaînage arrière pour permettre de travailler sur des buts, infirmer ou confirmer un diagnostic.

La deuxième version de CODIAPSY, qui va essayer d'aller un peu plus loin au niveau de la réflexion préalable, de tenir compte des critiques, d'inclure ce travail dans un cadre plus global, est en cours de réalisation.

De plus, parallèlement à CODIAPSY, un logiciel de dossier médical psychiatrique a été développé au GRIP. Il est destiné au suivi des patients et à la rédaction d'un résumé standard de sortie comportant évidemment une codification diagnostique. L'un des soucis étant de réduire les différences inter-cotateurs, l'autre, administratif celui-là, mais non négligeable, concerne l'uniformisation de la codification des paiements des soins.

L'idée actuelle est donc d'utiliser CODIAPSY, alimenté en faits directement à partir de la saisie des données résultant de la consultation et déjà notées sur le dossiers du patient, avec simplement et si nécessaire des demandes de compléments d'informations. A l'inverse les conclusions diagnostiques pourront être récupérées et apparaître dans le résumé standard de sortie. Ainsi l'utilisateur disposera d'une aide à la codificaition diagnostique parfaitement intégrée.

II esiste encore un autre objectif important pour le système, qui est d'offrir la possibilité aux autres classifications INSERM et CIM. Il y a là, pour les experts, un travail de recherche d'équivalences non négligeable, à faire en préalable.

CONCLUSION

En conclusion, nous soulignons l'importance d'une collaboration réussie avec mise au point d'une "technique" de conduite :

- pour une bonne collaboration entre équipes, il faut :
  * au moment de la prise de contact, définir les contraintes et le modus vivendi qui doivent être acceptés par tous,
  * mettre en place un calendrier précis des réunions,
  * exiger des rencontres régulières et respectées,
  * prévoir un ordre du jour dès la séance précédente, un compte rendu,
  * définir une tâche précise pour chacun entre les séances,

- pour une structuration du travail :
  * établissement d'un cahier des charges,
  * formulation/modélisation de la connaissance par les experts,
  * traduction par les experts et l'ingénieur de la connaissance,
  * mise au point,
  * validation globale avec retour aux étapes antérieures si nécessaire.

Dans le futur nous pensons intégrer le moteur d'inférence HAMEX dans la chaîne de traitement du dossier médical informatisé grâce à la création d'une version utilisateur permettant l'obtention d'un diagnostic automatique.

BIBLIOGRAPHIE

A. BARR, E.FEIGENBAUM
The handbook of artificial intelligence
Los Altos CA William KAUFMANN, inc (1982)

A.BONNET
Quelques modes de représentation des connaissances et des mécanismes de raisonnement pour les systèmes experts
4ième congrès : Reconnaissance des formes et intelligence artificielle. Roquencourt (1984)

B.BUCHANAN, R. DUDA
Principles or Rule-Based Expert Systems
Heuristic programming project report nxHPP 82-14
dept of computer science, Stanford University (82)

E. CHOURAQUI, H. FARRENY, D. KAYSER, H. PRADE
Modélisation du raisonnement et de la connaissance
TSI vol 4, nx4, (1985)

A. COLMERAUER, H. KANOUI, R. PASERO, P. ROUSSEL
Un système de communication homme-machine en français
Rapport de recherche Université Aix-Marseille II (1973)

M. DAVIS
Interactive transfer of expertise : acquisition of new inference rules.
Proceedings of the fifth international joint conference of artificial intelligence IJCAI (1977)

J. FARGUES
Contribution à l'étude du raisonnement, application à la médecine d'urgence
Doctorat d'état Paris VI (1983)

M. FIESCHI
Aide à la décision en médecine : le système Sphinx
Thèse d'état en médecine Marseille (1983)

W. GALE
Student phase 1- A report on work in progress
Workshop of A.I. and statistics April 85
Princeton (1985)

J.P. HATON
Intelligence artificielle en compréhension automatique de la parole T.S.I. nx3 (1985)

F. HAYES-ROTH
Knownledge-based espert systems : the state of the art in U.S. Pergamon infotech Report (1984)

M. LAURENT
La structure de contrôle dans les systèmes experts
TSI nx3 (1984)

M. MINSKI (1975)
The psychology of computer vision, Mc Graw Hill

R. SCHANK, R. ABELSON
Scripts, plans and knowledge
4ème IJCAI, p.151 à 157 (1975)

INDEX DES AUTEURS
AUTHOR'S INDEX