PROCEEDINGS
CSCSI/SCEIO

Conference
1984

The University of Western Ontario
London, Ontario, Canada
15, 16, 17 May 1984

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

COMPTE-RENDU DE LA CINQUIEME CONFERENCE BIENNALE
DE LA
SOCIETE CANADIENNE DES ETUDES D'INTELLIGENCE
PAR ORDINATEUR

THE UNIVERSITY OF WESTERN ONTARIO
LONDON CANADA

15-17 May/mai 1984

Canadian Society for Computational Studies of Intelligence
Société Canadienne pour Etudes d'Intelligence par Ordinateur

Copies of these Proceedings may be obtained by prepaid order to:

CIPS National Office
5th Floor
243 College Street
Toronto, Ontario
M5T 2Y1

Prepaid price: $25.00

A Message From the General Chairman/Un Message du President Général

In the pages which follow you will find the papers, both invited and refereed, which will be presented at this, the Fifth National Conference of the Canadian Society for Computational Studies of Intelligence. A conference however is much more than a set of paper presentations. It is a meeting of minds - not all of which are open! It is a place for meeting old friends and making new ones. It is a place for discussing issues and, usually at great length, non issues. It is a place for riding hobby horses and, occasionally, Derby winners. It is something like an academic circus, with its barkers and hucksters and three card men; its chamber of horrors and its hall of mirrors and, yes, its strong man and the fat lady. You have paid for your ticket (I hope!) and the rides are free; enjoy yourselves.

This particular conference of the Society is being held at a time when the subject matter which draws us together is passing through a period of high visibility. For the first time in the lives of most of us, industry, the Professions and yes, even Government want to know, if not us, then experts in our area of interest. Some of us, (though we're a dying breed), have been here before. I'm not sure that I like it and I'm not sure that I don't. As a one time bank manager T.S. Elliot said "this above all is the greatest treason, to do the right deed for the wrong reason." You could well transpose this, and anyway, you all know what happened to Beckett.

But enough of this amateur philosophising: there are really intellectually exciting things happening out there for those who can stand the pace. Whether we have to wait five or five to the fifth years for the first all singing dancing robot, remember that it is said to be better to travel hopefully than to arrive. I am not sure that I belive this last aphorism, but anyway, take a step or two along the way at The University of Western Ontario in London - home of the inaugural meeting of the Society more years ago than I care to remember.

Ted Elcock
General Chairman
Fifth National Conference
of the CSCSI/SCEIO

Dans les pages suivantes, vous trouverez les articles, invités et examinés, qui seront presentés à la Cinquième Conférence Nationale de la Société Canadienne pour Etudes d'Intelligence par Ordinateur. Pourtant une conférence est beaucoup plus qu'une série de présentation d articles. C'est une rencontre d'esprits - pas tous ouvert! C'est une place pour rencontrer de vieux amis et en faire de nouveaux. C'est une place pour discuter des questions et habituellement, aussi pour discuter longuement des sujets de peu de conséquence. C'est une place pour chevaucher ses dadas favoris et, parfois, des gagnants du Derby. C'est un peu comme un cirque académique, avec ses bonimenteurs et colporteurs et les hommes de trois cartes; sa chambre d'épouvante et sa maison de miroirs et, oui, son homme fort et sa femme grasse. Vous avez payé pour votre billet (je l'espère) et les tours sont gratuits; amusez-vous bien.

Cette conférence particulière de la Société a lieu à un temps où le sujet qui nous assemble passe par une period d'haute visibilitée. Pour la première fois dans la vie de la plupart de nous, l'industrie, les Professions et oui, même le Gouvernement veulent nous en savoir, si pas nous, alors les experts dans nos domains d intérêts. Quelques-uns de nous l'avions déjà vécu (malgré que nous sommes une espèce qui est en train de disparaître). Je ne suis pas certain que je l'aime mais je ne suis pas certain que je ne l'aime pas. Comme un ancien directeur d'une banque T.S. Elliott a dit "this above all is the greatest treason, to do the right deed for the wrong reason." Vous pouvez facilement transposer ceci, et en tout cas, vous savez ce qui est arrivé à Beckett.

Mais assez de ce philosophie amateur: Il y a des choses vraiment intellectuellements passionnantes qui arrivent pour ceux qui peuvent survivre les développements. Que nous ayons besoin d'attendre cinq ou cinq à la cinquième années pour le premier robot tout chantant et dansant, souvenez-vous qu on dit que c'est mieux de voyager avec espoir que d'arriver. Je ne suis pas certain que je crois ce dernier aphorisme, mais en tout cas, prenez un pas ou deux sur le chemin à l'Université de Western Ontario à London - place d'origine de la conférence inaugural de la Société plus d'année que j'aime me rappeler.

Ted Elcock
Président Général,
Cinquième Conférence Nationale
de la CSCSI/SCEIO

OFFICERS OF THE CSCSI/SCEIO

	1983-84	1984-85
President:	Nick Cercone Department of Computer Science Simon Fraser University Burnaby, B.C. V5A 1S6	Gordon McCalla Dept. of Computational Science University of Saskatchewan Saskatoon, Sask. S7N 0W0
Vice-President:	Gordon McCalla Dept. of Computational Science University of Saskatchewan Saskatoon, Sask. S7N 0W0	John Tsotsos Department of Computer Science University of Toronto Toronto, Ontario M5S 1A4
Treasurer:	Wayne Davis Department of Computing Science University of Alberta Edmonton, Alberta T6G 2H1	Wayne Davis Department of Computing Science University of Alberta Edmonton, Alberta T6G 2H1
Secretary:	John Tsotsos Department of Computer Science University of Toronto Toronto, Ontario M5S 1A4	Mike Bauer Department of Computer Science The University of Western Ontario London, Ontario N6A 5B7

OFFICERS OF THE FIFTH BIENNIAL CONFERENCE OF THE CSCSI/SCEIO

Program Chairman:

John Tsotsos
Department of Computer Science
University of Toronto
Toronto, Ontario. M5S 1A4

General Chairman:

Ted Elcock
Department of Computer Science
The University of Western Ontario
London, Ontario. N6A 5B7.

Local Arrangements Chairman:

Mike Bauer
Department of Computer Science
The University of Western Ontario
London, Ontario. N6A 5B7

Local Arrangements Committee:

Sandy Alfs, Department of Computer Science, UWO
Mary Mac Vicar, Department of Computer Science, UWO

Program Committee:

Ron Brachman, Fairchild R&D
Jaime Carbonnell, Carnegie-Mellon University
Veronica Dahl, Simon Fraser University
Randy Goebel, University of Waterloo
Allen Hansen, U. Massachusetts at Amherst
John Hollerbach, Massachusetts Inst. of Technology
Takeo Kanade, Carnegie-Mellon University
Victor Lesser, U. Massachusetts at Amherst
Drew McDermott, Yale University
Tom Mitchell, Rutgers University
John Mylopoulos, University of Toronto
Ray Perrault, Stanford Research Institute
Harry Pople, University of Pittsburgh
Zenon Pylyshyn, University of Western Ontario
Stan Rosenschien, Stanford Research Institute
Ed Stabler, University of Western Ontario
Bonnie Weber, University of Pennsylvania
Bob Woodham, University of British Columbia

In addition to the people listed on the previous page, special thanks is extended to several others without whom this conference would not have been possible:

Francine and Paul Rochefort for the French translations;

Jo Moore, Laura Pineault and Francine Rochefort for their assistance in the mammoth task of label production, envelope stuffing and mailing;

U.W.O. Graphic Services, in particular Alex Hamilton for his assistance with both the brochure and Proceedings, and Bob Engel and Bruce Maslen for their work on the Proceedings;

Jim McArthur, Kathy Bates, Janice Steels, Carol Guard and Ann Toth for coordination of arrangements for Delaware Hall and other U.W.O. facilities;

Peter Hoddinott and Dave Wyatt for their assistance with last minute details.

Financial contributions from the Natural Sciences and Engineering Council, Faculty of Science at The University of Western Ontario, and the CSCSI/SCEIO are gratefully acknowledged.

INVITED TALKS

"Computational Linguistics = Generalized Unification + Applied Graph Theory"
Martin Kay (XEROX PARC)

"Optical Phenomena in Computer Vision"
Steven Shafer (Carnegie Mellon)

"Robotic Manipulation"
Matthew Mason (Carnegie Mellon)

"The Future of Logic Programming"
Alan Robinson (Syracuse U)

"The Role of Physiology in Medical Reasoning"
Ramesh Patil (MIT)

"A Fundamental Trade-off in Knowledge Representation and Reasoning"
Hector Levesque (Fairchild R&D)

TABLE OF CONTENTS

NATURAL LANGUAGE

"Computational Linguistics = Generalized Unification + Applied Graph Theory"
Martin Kay (XEROX PARC)....1

"A Theory of Discourse Coherence for Argument Understanding"
Robin Cohen (U of Toronto)....6

"Scalar Implicature and Indirect Responses to Yes/No Questions"
Julia Hirschberg (U of Pennsylvania)....11

"Generating Corrective Answers by Computing Presuppositions of Answers, Not of Questions, or Mind your P's, not Q's"
Robert Mercer, Richard Rosenberg (U of British Columbia)....16

"Good Answers to Bad Questions: Goal Inference in Expert Advice-Giving"
Martha Pollack (U of Pennsylvania)....20

COGNITIVE MODELLING AND PROBLEM SOLVING

"Using Spreading Activation to Identify Relevant Help"
Adele Howe (ITT), Timothy Finin (U of Pennsylvania)....25

"Temporal Reasoning with Metric Constraints"
Thomas Dean (Yale U)....28

COMPUTER VISION I

"Optical Phenomena in Computer Vision"
Steven Shafer (Carnegie Mellon)....33

"Procedural Adequacy in an Image Understanding System"
Jay Glicksman (Texas Instruments)....44

"The Local Structure of Image Discontinuities in One Dimension"
Yvan Leclerc, Steven Zucker (McGill U)....50

"Receptive Fields and the Reconstruction of Visual Information"
Steven Zucker (McGill U), Robert Hummel (New York)....58

ROBOTICS

"Trajectory Planning Problems, I:Determining Velocity Along a Fixed Path"
Kamal Kant, Steven Zucker (McGill U).........................62

"Interpreting Range Data for a Mobile Robot"
Stan Letovsky (Yale U).......................................70

LEARNING

"The Use of Causal Explanations in Learning"
David Atkinson, Steven Salzberg (Yale U).....................73

"Experiments in the Automatic Discovery of Declarative and Procedural Data Structure Concepts"
Mostafa Aref, Gordon McCalla (U of Saskatchewan)............79

"Theory Formation and Conjectural Knowledge in Knowledge Bases"
James Delgrande (U of Toronto)...............................87

"Conceptual Clustering as Discrimination Learning"
Pat Langley, Stephanie Sage (Carnegie Mellon)...............95

"Some Issues in Training Learning Systems and an Autonomous Design"
David Coles, Larry Rendell (U of Guelph)....................99

"Inductive Learning of Phonetic Rules for Automatic Speech Recognition"
Renato De Mori (Concordia U), Michel Gilloux
(Centre National d'Etudes des Telecommunications,
France)...103

COMPUTER VISION II

"Applying Temporal Constraints to the Problem of Stereopsis of Time-Varying Imagery"
Michael Jenkin (U of Toronto)...............................106

"Scale-Based Descriptions of Planar Curves"
Alan Mackworth, Farzin Mokhtarian (U of British
Columbia)..114

LOGIC PROGRAMMING

Implementing PROGRAPH in Prolog: An overview of the Interpreter and Graphical Interface"
Philip Cox, Tomasz Pietrzykowski (Acadia U)................119

"Making 'Clausal' Theorem Provers 'Non-Clausal'"
David Poole (U of Waterloo)..................................124

"Logic as an Interaction Language"
Martin van Emden (U of Waterloo).............................126

EXPERT SYSTEMS AND APPLICATIONS

"ROG-O-MATIC: A Belligerent Expert System"
Michael Mauldin, Guy Jacobson, Andrew Appel, Leonard Hamey (Carnegie Mellon)....................................129

"An Explanation System for Frame-Based Knowledge Organized Along Multiple Dimensions"
Ron Gershon, Yawar Ali, Michael Jenkin (U of Toronto)..133

"Qualitative Sensitivity Analysis: An Application to Expert System Plan Justification"
Stephen Cross (Air Force Institute of Technology)..........138

KNOWLEDGE REPRESENTATION

"A Fundamental Trade-off in Knowledge Representation and Reasoning"
Hector Levesque (Fairchild R&D)..............................141

"Representing Control Strategies Using Reflection"
Bryan Kramer (U of Toronto)..................................153

"Knowledge Base Design for an Operating System Expert Consultant"
Stephen Hegner (U of Vermont), Robert Douglass (Los Alamos National Laboratory)............................159

"Steps Towards a Theory of Exceptions"
James Delgrande (U of Toronto)...............................162

CSCSI/SCEIO SURVEY

"Canadian Artificial Intelligence in the Next Decade Funding, Strategies, and the Role of the CSCSI/SCEIO"
Nick Cercone - President CSCSI/SCEIO
Gordon McCalla - Vice-President CSCSI/SCEIO...............167

Computational Linguistics = Generalized Unification + Applied Graph Theory

Martin Kay

Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto
California 94304

Introduction

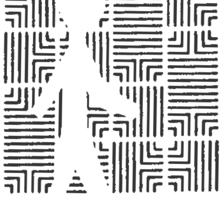
We usually characterize things from on top—from the point of view of how they fit into some larger picture. So, if someone asks me what my claim to be a computational linguist implies for the way I spend my time, I might be expected to wax eloquent about building computational models to help explain deep psychological phenomena or at least about the importance of machine translation for world peace. My aim in this paper is to characterize computational linguistics from below, say from the point of view of a tool maker trying to decide where he might best invest his ingenuity.

I shall begin with an operation that I call *generalized unification* which applies to sets of descriptions. If there could be some objects that all the descriptions refer to, then the operation delivers a single description of this set. If there are none, the operation is said to fail. Much of the data that linguists work with can profitably be cast in terms of descriptions of the appropriate kind, in which case many computational operations that had previously seemed unrelated, reduce to unification. However, the familiar notion of unification needs to be generalized, in particular, in order to allow it to treat different types of data in different, though mutually consistent ways. Much of the data can usefully be represented in the form of directed acyclic graphs with labeled nodes and arcs. One particular type of data that has an obvious place in linguistics consists of sets of strings, in particular, regular sets. Regular sets are readily characterized by finite automata which, in turn, are routinely represented by directed graphs. A programming system that implemented generalized unification and the principal operations of graph theory, particularly as it relates to finite automata, would therefore meet the needs of computational linguists belonging to a wide variety of theoretical sects.

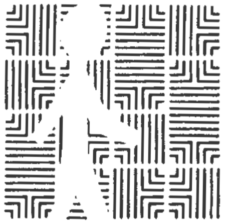
I will begin with an informal account of generalized unification, motivating it with examples from international espionage, syntax and semantics. This will lead naturally to a discussion of the roles played by finite automata in syntax and morphology as well as to a related kind of object called a *parsing chart*.

Generalized Unification

Suppose that, in your capacity as Director of Intelligence for the Department of Fine Arts, you send one of your operatives to photograph an important design on a wall in some far-off place. What you receive back is these two microfilm frames, now publicly displayed for the first time.



Apparently, someone was walking in front of the design when both photographs were taken. After weeks of feverish work, the boys in the green eyeshades come up with a new technique, now also revealed for the first time. They make transparencies from the photographs and, when they project them superimposed on one another, the intruding figure disappears. The technique they discovered is what we now know as *unification*. It consists in creating a picture from two or more potentially incomplete originals, filling holes in each with information taken from corresponding places in the others. The result may, of course, also be incomplete, but since it contains the union of the information in the contributors, it produces a result that is at least as good as they are. In the course of unification, it may be discovered that a pair of pictures are deceptively similar but that they could in fact not be of the same scene. The pictire below, for example, could not be the same wall because there is a different pattern of lines in the top lefthand square. In this case, unification fails.



The microfilm story is, of course, hopelessly contrived. Even if the registration of the two images were perfect, the person that obscured part of each would probably not have been transparent. Furthermore, this latest picture could in fact depict the same scene as the earlier one if we continue to insist that the black parts of the images contain reliable information whereas the white parts contain no information at all. What this means, however, is that unification alone would probably not be sufficient to the needs of the intelligence agency, not that the notion of unification itself has been betrayed in any fundamental way.

Consider another example, also from the world of cloak and dagger. This time, we are comparing the descriptions given below with a view to determining whether they might be of the same person. We cannot be sure that they are the same, but nothing contradicts that hypothesis. In order to be sure of this, we need to be able to convert between pounds and kilograms and to know that a 48-year-old person is middle aged. We know that we need not be too dismayed by the apparent disparity in children's names because, while we expect the information we are given to be correct, we know that it is not complete. On the assumption that the people are the same, we can formulate a new description in the obvious way which would be more complete that either of the originals but necessarily true of the person if the originals were true. It does not matter what units we choose to represent the lady's weight. The figure 48 is presumably better than middle-aged on the grounds that it is more accurate. Our new view is of a man with brown hair because we know to combine propositional expressions using the rules of propositional logic. Our new composite picture will be of a man with at least three children, Ahmed, Rebecca, and Angela. Notice that, while we are not directly interested in the wife, the descriptions we bave of her appear as part of the description of the man and unifving them occurs as part of the process of unifying tbe descripions of the man. An inconsistency there would cause the unification to fail just as if it had occurred among the man's primary attributes

eyes: blue	eyes: blue
hair: black or brown	hair: brown or red
height: 5'11"	height: 5'11"
accent: Italian	
	name: O'Grady
wife: *see below*	wife: *see below*
children: Ahmed, Angela	children: Rebecca, Angela
age: middle	age: 48

The wife:
eyes: brown
weight: 247lbs 112.015 Kg.
disposition: surly

This has been an exercise in *generalized* unification, which differs from ordinary unification in two respects. First, the set of attributes that can be part of a description is open ended. New reports can always bring new information and there is therefore no sense in which a description can ever be said to be complete. Second, the ways in which the values of attributes are compared differs with the type of the value. For example, metric conversion can be applied to weights, set union to the names of children, and the unification process itself to embedded descriptions. In logic programming, where most of us probably gained whatever familiarity with unification we may have, there are just three kinds of entities, constants, variables, and expressions. The variables are the holes in our picture; the constants and expressions are the solid object that can fill them. But if a particular variable occurs more than once in the expressions being unified, each occurrence must be filled in the same way. If A, B, and C are constants and x and y are variables, we may take $(AyC)B)$ and (xy) to be typical expressions over them. When they are unified, B becomes the value of the variable y, filling that hole, and (ABC) becomes the value of x.

Attribute-value lists, with values of various types, are a natural form in which to couch descriptions of objects in general and of linguistic objects in particular. It is easy to see how a dictionary entry might be put in this form. The structure of a sentence can be represented simply and perspicuously using such attributes as "Subject", "Indirect object", "Tense", "Topic", "Gender", "Predicate", and "Argument". As we shall now see, the grammar of a whole language, which is a description of the set of expressions that it contains can also be expressed in this form. If this is done, and if the rules for comparing and combining various kind of information are defined appropriately, then unification, rather than string manipulation, emerges as the primary operation of linguistic computation.

Syntax

There are several modern syntactic theories in which unification, or something equivalent to it plays a crucial role. Among these are Lexical Functional Grammar, Functional Unification Grammar, Generalized Phrase Structure Grammar, and the PATR grammar currently under development at SRI International. Since the details that distinguish these formalisms are of no concern to my present aim, I will illustrate my point with a formalism that is different from all of them except in its use of unification. We start with five familiar rules of context free grammar, namely

1. S ⟶ NP VP
2. VP ⟶ V NP
3. VP ⟶ V
4. NP ⟶ Det N
5. NP ⟶ N

and write their counterparts in the new formalism. Rather than explaining the formalism in detail, I append annotate each rule with a short commentary.

$$\begin{bmatrix} \text{Cat} = \text{S} & \\ \text{Head} = & \begin{bmatrix} \text{Cat} = \text{VP} \\ \text{Head} = [\text{Subj} = [\text{Cat} = \text{NP}]] \end{bmatrix} \end{bmatrix}$$

$$\longrightarrow \quad <\text{Head Head Subj}> \; <\text{Head}>$$

The label, or formal description, of a sentence must have at least the attributes *Cat* and *Head*. Any other attributes that it may acquire for any reason, are irrelevant. The *Cat* attribute has the value S and the head has as its value a subexpression, or embedded description, in which the attributle *Cat* has the value VP. This has a *Head* attribute whose value has a *Subj* attribute, and this has the description $[Cat = NP]$. So much for the label. A sentence has two parts, or constituent phrases, whose descriptions are embedded in that of the sentence itself. The value of the first is the value of the *Subj* attribute of the value of the *Head* attribute of the value of the *Head* attribute of the sentence, or its Head's Head's Subject. The second constituent is the value of the *Head* attribute of the sentence: its Head. Notice that the description of the sentence properly contains the descriptions of its constituents, but one of these properly includes the other.

$$\begin{bmatrix} \text{Cat} = \text{VP} & \\ \text{Head} = & \begin{bmatrix} \text{Cat} = \text{Verb} \\ \text{Obj} = [\text{Cat} = \text{NP}] \end{bmatrix} \end{bmatrix} \longrightarrow \quad <\text{Head}> \; <\text{Head Obj}>$$

A phrase whose description has the attribute *Cat* with the value VP (a verb phrase) has two constituents, one its head and the other its Head's Object. The former must be describable as $[Cat = V]$ and the latter $[Cat = NP]$. Once again, the description of the object is *part of* the description of the verb.

$$\begin{bmatrix} \text{Cat} = \text{VP} & \\ \text{Head} = & \begin{bmatrix} \text{Cat} = \text{Verb} \\ \text{Obj} = \text{NONE} \end{bmatrix} \end{bmatrix} \longrightarrow \quad <\text{Head}>$$

A verb phrase is not required to have an object and, in this case, rule 3 applies in place of rule 2. The Head's Object has the value NONE and the Head is the only constituent.

$$\begin{bmatrix} \text{Cat} = \text{NP} & \\ \text{Head} = & \begin{matrix} \text{Cat} = \text{Noun} \\ \text{Art} = [\text{Cat} = \text{Det}] \end{matrix} \end{bmatrix} \longrightarrow \quad <\text{Art Head}> \; <\text{Head}>$$

Rule 4 is like rule 2;

$$\begin{bmatrix} \text{Cat} = \text{NP} & \\ \text{Head} = & \begin{bmatrix} \text{Cat} = \text{Noun} \\ \text{Art} = \text{NONE} \end{bmatrix} \end{bmatrix} \longrightarrow \quad <\text{Head}>$$

And rule 5 is like rule 3.

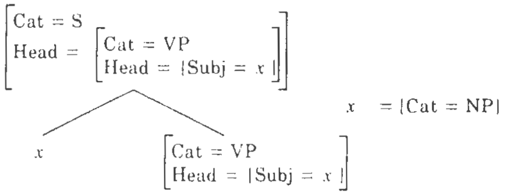
Needless to say, this is a trivial grammar but it will serve the purposes of illustration. Let us now consider how this grammar might be used in the generation of a simple sentence. Analogously with the rewriting procedure used with context-free grammars, we begin by writing down a particular expression, namely that which appears to the left of the arrow in the first rule. This will label the top node in the tree we shall grow. The rule tells us that there will be two nodes beneath it. The occupant of the first of these must be describable as $[Cat = NP]$ and what occupies the second as

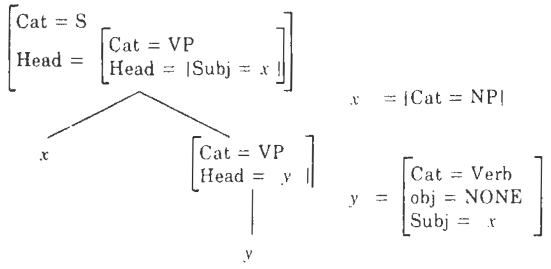
$$\begin{bmatrix} \text{Cat} = \text{VP} \\ \text{Head} = [\text{Subj} = [\text{Cat} = \text{NP}]] \end{bmatrix}$$

But there is a little more information in the rules than this suggests and, to and it is revealed in the tree diagram at the head of the next column: The variable x, with the value $[Cat = NP]$ has been used to indicate the fact that whatever description occupies one of the sites labeled x in the tree must also appear at all the others. A rule must be found to apply to the $[Cat = VP...]$. It must be one whose left-hand side is a description that is unifiable with the label on the node and the candidates are clearly rules 2 and 3. Let us take rule 3, thus committing ourselves to a single new node on the right labeled

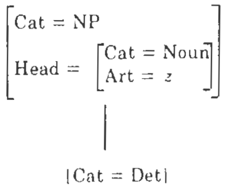
$$\begin{bmatrix} \text{Cat} = \text{Verb} \\ \text{Object} = \text{NONE} \end{bmatrix}$$

We now have the following tree structure shown at the foot of this column. The bottom node on the right is a terminal node in the





usual sense that there are no rules that can be applied to it. The left-hand node can be expanded by rule 4 or rule 5 and we will take rule 4 giving us the following value for x.



So far, the complexity we have added to the familiar context-free grammar will doubtless seem gratuitous. Its cash value will begin to appear when we begin to examine the lexical descriptions of the items that can fill the terminal positions in this structure. Let us begin with the first word. Its description in the lexicon must be one that unifies with $[Cat = Det]$. Possible examples might be:

$$\begin{bmatrix}\text{Cat = Det}\\ \text{Word = the}\end{bmatrix}\quad\begin{bmatrix}\text{Cat = Det}\\ \text{Word = this}\\ \text{Num = sing}\end{bmatrix}\quad\begin{bmatrix}\text{Cat = Det}\\ \text{Word = these}\\ \text{Num = Plur}\end{bmatrix}\begin{bmatrix}\text{Cat = Det}\\ \text{Word = all}\\ \text{Num = Plur}\end{bmatrix}$$

"This" and "These" are marked to show that they are singular and plural. "The" is not marked because it can be used in either a singular or a plural context. Some nouns that might occupy the second position in our sentence are

$$\begin{bmatrix}\text{Cat = Noun}\\ \text{Word = dog}\\ \text{Art = [Num = sing]}\end{bmatrix}\quad\begin{bmatrix}\text{Cat = Noun}\\ \text{Word = dogs}\\ \text{Art = [Num = plur]}\end{bmatrix}$$

$$\begin{bmatrix}\text{Cat = Noun}\\ \text{Word = Fido}\\ \text{Art = NONE}\end{bmatrix}\qquad\begin{bmatrix}\text{Cat = Noun}\\ \text{Word = sheep}\end{bmatrix}$$

The crucial point to notice is that the description of the article must be unifiable with the Art attribute of the noun. If we choose the first of these, the variable v would be required to have the value *NONE*, conflicting with the one already assigned in rule 4. Apparently, the noun "Fido" can only be used in a noun phrase that has no determiner. For similar reasons, the other three nouns can be used only in a noun phrase that does have a determiner. Now, if the word "dog" is chosen, v must have the property $[Num = sing]$, restricting the choice of determiner to "the" and "this". Notice that these agreements are not specified in detail by the grammar: all that is required is that the description of the head of the phrase contain a description of the determiner, thus enabling it to restrict the determiner in arbitrary ways.

The verb is taken to be the head of the sentence and it can impose arbitray restrictions on all the other constitutents. The following are lexical entries for some possible verbs

$$\begin{bmatrix}\text{Cat = Verb}\\ \text{Word = devours}\\ \text{Obj = [Cat = NP]}\\ \text{Subj = [Num = sing]}\end{bmatrix}\begin{bmatrix}\text{Cat = Verb}\\ \text{Word = devour}\\ \text{Obj = [Cat = NP]}\\ \text{Subj = [Num = plur]}\end{bmatrix}$$

$$\begin{bmatrix}\text{Cat = Verb}\\ \text{Word = slept}\\ \text{Obj = NONE}\end{bmatrix}\begin{bmatrix}\text{Cat = Verb}\\ \text{Word = ate}\end{bmatrix}$$

If we choose "devours" as the verb in our sentence, we are committed to a value for the variable x with the property $[Num = sing]$, whereas if we choose "devour", we are committed to one with the property $[Num = plur]$. It goes without saying that we are simplifying the facts of English grammar immensely, in particular, by restricting ourselves to third-person forms. However, it should be clear that person can be treated in an entirely analogous manner. "Devour" is a transitive verb and accordingly can only be usedin a verb phrase whose *Obj* attribute has the value $[Cat = NP]$, whereas "slept" is intransitive and does not allow an object. Being a past tense form, "slept" also places no restriction on the number of its subject. Finally, the lexical entry for the verb "ate" places no restrictions either on its subject or on its object because it is a past tense form that can be used either transitively or intransitively.

There are two important features that set the family of formalisms of which this is a member off from its predecessors in computational linguistics, like Augmented Transition Networks and in computer science, such as Attribute Grammars. One is that there are entirely declaritive not requiring grammar writers to know anything about a particular sequence of events that will be followed either in generation or in analysis. The other is that the only operation that needs to be added to what is already known about context-free grammars to achieve this is that of generalized unification. From the declaritive property, there follows a considerahle amount of computational robustness; in particular, we have a formalism with the kind of power that seems to be required to describe natural languages in a theoretically revealing way while supporting the computation operations of generation and analysis with equal facility.

Semantics

I now hope to show that the benefits of unification go well beyond those just outlined. Let us augment the lexical entries for "all" and "dogs" so that they become as shown at the top of the next column. Observe what will happen now when, following rule 4, these two words are incorporated in a noun phrase. The determiner's description is unified with the value of the *Art* property of the noun, causing the variable p to take on the value of he *Meaning* property of the determiner unified with the value of the meaning property of the noun. This unification gives the following result:

$$\begin{bmatrix}\text{Type = all} & \\ \text{Var = } q & \\ \text{Prop = } & \begin{bmatrix}\text{Type = and} & \\ \text{P1 = } & \begin{bmatrix}\text{Pred = dog}\\ \text{Arg = } q\end{bmatrix}\\ \text{P2 = [Arg = } q\text{]} & \end{bmatrix}\end{bmatrix}$$

Without going into elaborate detail, this can be seen as description of the logical expression

$$\forall q.dog(q)\wedge P(q)$$

The predicate P remains to be specified. It would come to be specified if the noun phrase "all dogs" hecame the subject of the verb "ate", and the lexical entry of "ate" were provided with a *Meaning* as follows:

$$\begin{bmatrix}\text{Cat = Verb}\\ \text{Word = sleep}\\ \text{Meaning = } r\\ \text{Subj = [Meaning = } r\text{ = [Prop = [P2 = [Pred = eat]]]}\end{bmatrix}$$

The meaning of the verb is unified with that of the subject, and P2 of the Prop of that meaning is unified with [Pred = eat]. It is not

$$\begin{bmatrix} \text{Cat} = \text{Det} \\ \text{Word} = \text{all} \\ \text{Num} = \text{plur} \\ \text{Meaning} = \begin{bmatrix} \text{Type} = \text{all} \\ \text{Var} = q \\ \text{Prop} = \begin{bmatrix} \text{Type} = \text{Implies} \\ \text{P1} = [\text{Arg} = q\,] \\ \text{P2} = [\text{Arg} = q\,] \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{Cat} = \text{Noun} \\ \text{Word} = \text{dogs} \\ \text{Art} = \begin{bmatrix} \text{Num} = \text{plur} \\ \text{Meaning} = p \end{bmatrix} \\ \text{Meaning} = p \quad = \begin{bmatrix} \text{Prop} = \begin{bmatrix} \text{P1} = \quad \begin{bmatrix} \text{Type} = \text{Pred} \\ \text{Pred} = \text{dog} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

difficult to see that the result of this is effectively

$$\forall q.dog(q) \wedge eat(q)$$

Grammars belonging to the family we have heen considering have several clear advantages over context-free grammars. It is possible to show that they are more powerful as measured in the usual way, namely by the size of the class of sets of strings that it is possible to characterize with them. This may he more or less closely related to the more important fact that they can be used to characterize natural languages in a more concise and perspicuous manner. In addition, they make it possible to locate more of the constraints that characterizing a language entails in the lexicon, thus naturally accommodating in some measure the massive lexical idiosyncracy that seems to be characteristic of natural languages. Furthermore, these formalisms have the facilities necessary to establish relationships among different kinds of structure, as we illustrated with the most recent example.

A common characteristic of these grammars is that they impose two different structures on each sentence, one a *constituent structure* which is a tree such as has long heen familiar, and the other a *functional structure*, which is a recursive structure of attributes and associated values. In general, this is not a tree but a *directed acyclic graph*. If it were a tree, then we should not have found it necessary to use variables to describe this structures. Variahles turn out to be necessary in just those cases where a node has more than one parent, thus violating one of the principal conditions that a tree must meet.

Word Order and Configurationality

In respect of their ability to state ordering relationships among the words and phrases in a sentence and to capture significant generalizations that these relationships exhibit, these grammars share many of the difficiencies of context-free grammar. In particular, while they are able to characterize so-called free-word-order and nonconfigurational languages clumsily at best. A pure free-word order language would be one with the property that the members of any phrase can be arbitrarily rearranged without substantantially changing its meaning or its grammatical properties. There are no pure examples of such languages, but there are many that come close. A pure nonconfigurational language would be one in which the members of a syntactic phrase do not have to be adjacent to one another. These are also not found in the pure form. On the other hand, many more familiar languages seem to obey ordering constraints of a stronger kind than can be readily stated in an unadorned context-free grammar. For example, it may be that whenever a verb and a noun appear in the same phrase, the verb invariably precedes the noun.

In response to such observations as these, the designers of some formalisms have felt impelled to introduce new mechanisms for describing order. Lexical Functional Grammar and Functional Unification Grammar both allow the string of elements that make up a phrase to be described by a regular expression. In the latter case, the expression can also designate places in the string where material from outside the phrase can be allowed to intrude. One can get some of the flavor of this by modifying the formalism we have been using in this paper in a similar direction. Consider the following rule:

$$\begin{bmatrix} \text{Cat} = \text{NP} \\ \text{Head} = x \quad = \begin{bmatrix} \text{Cat} = \text{Noun} \\ \text{Art} = [\text{Cat} = \text{Det}] \end{bmatrix} \\ \text{Mod} = \begin{bmatrix} \text{Cat} = \text{S} \\ \text{Binding} = \; x \end{bmatrix} \end{bmatrix}$$

$$\longrightarrow \; <\text{Head Art}> \; <\text{Head}> \; <\text{Mod}>$$

This is intended to be taken as meaning that a noun phrase may have three constituents: its head's article, the head itself, and a modifier. The modifier is a sentence with a non-null value for the *Binding* attribute and this will be realized as a relative clause through mechanisms that are beyond our scope. Following the list of constituents is a separate statement of the order in which it is permissible for them to occur, namely with the article and head adjacent to one another and in that order and with the modifer following, though not necessarily immediately. This allows for sentences like "A man came in whom I did not recognize". If the ordering information had been omitted altogether, the understanding would have been that the three parts of such a noun phrase could occur in any order and without any requirement that they be adjacent.

The right-hand side of the above rule has been broken into two parts, one a set of constituent names and the other a regular expression, with # as a "wild card" constraining their order. The hypothetical constraint mentioned earlier, that a verb always precedes a noun when both are members of the same phrase is stated in the following regular expression

$$\Sigma^* - [\Sigma^* \text{ noun } \Sigma^* \text{ verb } \Sigma^*]$$

This set contains all strings over the alphabet Σ not containing a member of the offending set

$$[\Sigma^* \text{ noun } \Sigma^* \text{ verb } \Sigma^*]$$

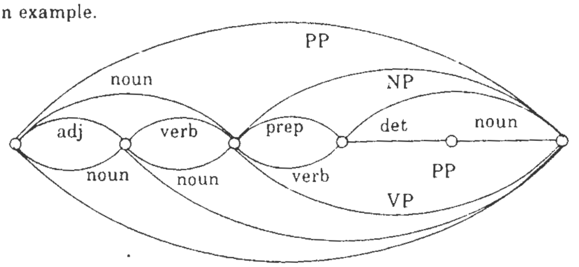
Non-trivial computation with regular sets must invariably be carried on the in equivalent domain of finite-state machines, and computation with finite-state machines is effectively computation with the labeled directed graphs whose nodes and arcs represent their states and transitions. So it seems that there are directed graphs with at least three different interpretations that play crucial roles in computing with the new kinds of grammar. But there is at least one more.

The Parsing Chart

The sentences of natural language are notorious the number of different kinds of ambiguity that frequently occur in them. At one time or another, almost everybody has found it socially acceptable to show some amusement at sentences like "Time files like an arrow but fruit flies like a banana". In the absence of any *a priori* basis on which to prefer one syntactic alternative to another, still less to compute only the desirable alternative, computational linguists have tried to develop means of analyzing sentences that

1. Find all alternatives,
2. Find none of them more than once,
3. Reuse parts that can be incorprated in more than one structure rather than computing them once for each,
4. Are simple and perspicuous,
5. Leave the sequence of events as fluid as possible so that any heuristic that shows any likelihood of producing better alternatives earlier can be incorporated without changing the overall scheme,
6. Are relatively insensitive to minor differences in linguistic theory.

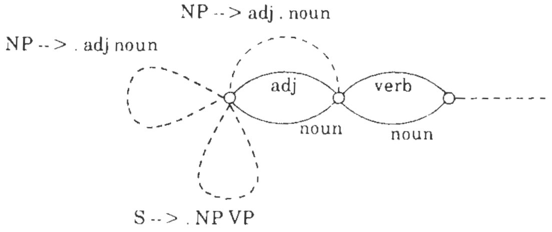
The strategy that seems to come closest to meeting all these requirements is based on the notion of a *parsing chart*. By this time it will be no surprise to learn that this is a directed graph. Here is an example.



The chart contains the information that results from parsing the setence "Time flies like an arrow" using a context-free with just those rules necessary to produce the ambiguity for which it is famous, but simplified somewhat for expository purposes.

There is a node in the chart for the beginning and end of the setence as well as for the space between each word. The grammatical labels assigned too each word and phrase are written against and edge that spans that appropriate segment of the sentence. If two or more edges have the same scope, then there are competeing interpretations for that part of the string. A realistic grammar, and in particular one of the kind we considered earlier, would label each edge with a recursive structure of attributes and values. The simple part-of-speech labels in this diagram can be thought of as standing in for these. In fact, they must be allowed to stand in for something still more complex because the label on each edge must also explicitely name the other edges that are labeled with its grammatical constituents. Only if this done will it be possible to recover constituent trees unambiguously from the chart.

A complete pasing chart contains and edge for every grammatically allowable word and phrase, just as the one we have exhibited does, and these are its active edges. It also contains active edges which represent partial phrases. The label of an active edge contains, aamong other things, a grammar rule together with an indication of which parts of it have already found a match. If the chart shown above is thought of as the final result of parsing the sentence "Time flies like and arrow", omitting active edges, then the following shows the state of affairs that might have obtained at an early stage in the parsing process, with the active edges included.



The edges closest to the horizontal axis of the diagram presumably result from a morphological analysis of the individual words and we assume that they are in place when parsing proper begins. We have shown active edges with dotted lines and have annotated them with a context-free rule with a dot somewhere among the symbols on its right-hand side. The symbols to the right of this dot have already found a match among the inactive edges in the chart; the symbol immediately to the right of the dot must find a match among the inactive edges incident from the node at which the active edge in question terminates. Otherwise, it represents a step in an unsuccessful attempt to apply the rule.

A complete rehearsal of the priciples of chart parsing are clearly beyond the scope of this paper But enough has already been said to establish it as an exercise in the manipulation of directed graphs. In fact, while there is presumably little profit in doing so, it is possible to interpret the inactive edges of a parsing chart as the transition diagram of a finite-state machine. The set of strings the generate is simply the lines in the various syntactic derivations of the sentence.

For a final example of the crucial role played be directed graphs in linguistic computing, we turn to morphology; more particularly morphopbonemics or morphographemics. A given lexical item—a stem, prefix or suffix—is often written or spelled differently depending on the context in which it occurs. The *o* in "telephone" is pronounced differently from the one in "telephonic" and the word "spy" changes its spelling when a plural "s" follows. For decades, linguists have relied no cascades of string rewriting rules to describe these phenomena. Given a sequence of items taken from the dictionary, say "spy" followed by "s", it is a simple matter to apply the rules in order to produce the word "spies". However, it is anything but a simple matter to reverse this process so as to obtain "spy+s" from "spies". It turns out that there is nothing intrinsically unidirectional about the rules and that it is possible to recast them automatically as finite-state transducers. Here, for example is a rule the handles the "spy/spies" case:

{{(Final-Y:i) {a:a | e:e | o:o | u:u | Mute-E:e | S-Suffix }
| (Final-Y:y) OTHER }* (Final-Y:y)}

It reads quite simply. An english word consists of a sequence of segments each of which is one of the following:

1. One of the letters "a", "e", "o", or "u" coming from the same letter in the dictionary, or an "e" coming from the speical dictionary character "Mute-E" or the "-s" suffix. Preceding these inthe segment, there may optionally be an instance of the dictionary character "Final-Y" representented in the text as "i".

2. Any dictionary-text combination not mentioned in this expression, possibly preceded by a "Final-Y" represented in the text as "y".

The foregoing notwithstanding, a word can end in a "y" corresponding to a "Final-Y" in the dictionary.

Conclusion

Efforts have been made in the past to provide computational linguists with specialized tools. Notable among these was the programming language COMIT which provided a rich set of primitives for manipulating strings. That was in the 1960's. In recent years, philanthropic activity of this kind has falled off because, I believe, there has been no clear perception of what kinds of tool would best serve the needs of our field. If I am right about what I have said here, a clearer picture is now beginning to emerge. Some of the benefits of actually building a programming environment based on generalized unification and applied graph theory would be the obvious ones of reducing the labor involved in writing programs and of increasing the quality and perspicuity of those programs. I believe that we should also benefit from a sharpened perception of the differences among the theories that inderlie the programs that we write. In particular, I believe that programming can do much to strip theories of inessential rhetorical trappings and, that if there were encouragement to do this in our field, there would be a great deal more unity than the present fragmentd scene suggests.

A THEORY OF DISCOURSE COHERENCE FOR ARGUMENT UNDERSTANDING

Robin Cohen
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4

Abstract

This paper describes a theory of coherence for argument structure, developed as part of a computational model for analyzing arguments. The theory of argument understanding aims to restrict the search for interpretation of propositions to a computationally reasonable task, but at the same time assure that the majority of possible argument structures used by the speaker will be recognized. This dual goal of efficiency and robustness can be achieved by presenting a characterization of coherent transmission forms and reducing analysis to a recognition of these forms (in the absence of additional clues from the speaker as to the intended structure). The limitations yield an analysis algorithm of linear complexity and capture the set of acceptable argument structures to be recognized. The proposed coherence theory also sets a framework for establishing interpretations for each of the propositions of an argument within an overall argument representation - an analysis issue largely ignored in other argument understanding research.

1. Goals of research

The theory of coherence for argument understanding described in this paper is one part of a general computational model for the analysis of arguments, developed in [Cohen 83]. The aim of this research is to model a patient listener in a conversational setting where the speaker is trying to convince the hearer of a particular point of view. For the model to be effective, it should exhibit two important properties:
(i) efficiency: the amount of time required to derive a representation for the argument should be computationally well-behaved.
(ii) robustness: a wide variety of possible input structures should be accepted.

In [Cohen 81] certain processing strategies are outlined as a basis for the model and efficiency measures demonstrated. The restrictions selected, however, are crucially dependent on a characterization of the possible forms of input used by the speaker. The purpose of this paper is to describe the proposed restrictions as a theory of coherence for arguments, and to claim that that a model which designs its analysis methods according to such a theory can successfully combine the two required goals of efficiency and robustness. An understanding of the overall goals of the research will serve to motivate the development of the coherence theory.

There have been relatively few efforts to study arguments as a specific form of input to natural language understanding systems (NLUS). Most argument understanding research to date, however, has focused on the problem of generating a response to an argument within a conversational setting ([Birnbaum et al. 80], [Reichman 81]). A necessary pre-requisite process is an analysis of the input, constructing a representation of the intended meaning, in order to select ideal avenues for rebuttal.

In [Birnbaum et al. 80] a representation for the argument is generated and several possible disagreements outlined for each point raised. The representation constructed indicates both support and attack for points raised, and shows the points raised by both conversants. It assumes a shared conception of the argument structure between speaker and hearer, used by each in the subsequent construction of dialogue. Variability in beliefs between speaker and hearer is only apparent in each conversant's selection of salient parts of the argument graph to address in generation.

It is our contention that the speaker may conceive of connections between his points which will not be recovered by the hearer, because of difference in beliefs. Consider the following example:

EX1: 1) The great white likes to tap dance
2) It is a shark

Suppose the speaker believes: 3) All sharks like tap dancing. From the speaker's point of view, 2) may serve as evidence for 1) using a modus ponens rule of inference where 3) and 2) together combine to lead to the conclusion of 1) (with "great white" instantiating the generalization about sharks for this particular case). But clearly, most hearers will not see any support relation between 1) and 2) in EX1, since a premise like 3) is not considered plausible to fill in as intended by the speaker.

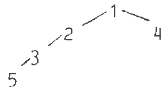
Because of cases like EX1, it is more appropriate to construct a representation for the argument from the framework of the hearer himself. Furthermore, the hearer should be able to record connections between propositions that he does not currently believe. But these will only be connections he feels could be plausible beliefs of another person, or if he knows the speaker well, plausible contentions of the speaker. Note that this problem of reconstructing argument structure is centred around the fact that most arguments have unstated premises which must be surmised by the other conversant. Our first main design decision is thus to build the representation of the argument from the framework of the hearer.

The second main design approach is to treat an argument as a set of propositions, and to then focus on the recognition of relations between propositions in an argument. Once more, the efforts of [Birnbaum et al. 80] and [Reichman 81] indicate in overall argument diagrams which propositions do relate and according to which relation (e.g. support, contrast). But there is no discussion of how an argument analyzer could take a stream of propositions uttered and successively build a representation by checking possible relations for each new proposition. Our approach is to specify rules of coherence to outline where each new proposition may fit with respect to the argument so far. As a result, we also bring to attention the issue of verifying possible relations between propositions, elaborating on the problems in defining relations such as "evidence".

The proposed analysis model now addresses a theory of argument coherence in that both relations between propositions and goals of the speaker are important. The theory of coherence for argument transmission can then be used to cover both overall aims of the model, as follows: (i) analysis can be restricted to the acceptance of a limited set of argument forms to ensure efficiency and (ii) robustness can be achieved by characterizing the set of acceptable transmissions, excluding transmissions with argument structures which could be deemed too difficult for a hearer to reconstruct. To briefly illustrate what constitutes a coherent transmission consider the following:

EX2:
1) The city is a mess
2) The parks are ruined
3) The benches are all rotting
4) The highways are all eroded
5) The legs can barely support the seat

In this example, one possible overall representation for the argument is a structure of claim and evidence relations as follows:



[The tree notation has sons as evidence for their father; the numbers correspond to the propositions of the argument; the representation serves to indicate the function of each proposition in the overall argument. (This is the notation used throughout [Cohen 83])]. Here, the speaker may intend that 5) serve as evidence for 3). But this is not a relation which a hearer should be expected to recover as part of a restricted search for coherent transmissions. In short, certain propositions will simply not be considered while a current proposition is being interpreted. Further discussion of this example appears in section 3, as the proposed restricted processing theory is outlined.

2. Overview of argument analysis model

Consider a computational model used to analyze an argument a proposition at a time, building a representation of the overall structure (claim and evidence relations). The model is divided into three main components:

a) Proposition Analyzer

This module takes the current proposition to be analyzed and a representation for the argument "so far" and assigns an interpretation to the proposition by including it in the argument representation, showing its relation to previous statements.

b) Linguistic Clue Interpreter

This module guides the analysis of a proposition in the presence of special words and phrases explicitly used by the speaker to indicate argument structure. For example, connectives such as "as a result" or "for example" provide some indication of the claim and evidence relation between the connecting propositions. The clue interpreter thus constrains the operation of the proposition analyzer.

c) Evidence Verifier

To build the overall representation, the test for evidence relations between propositions - e.g. "is A evidence for B?" is separated and delegated to this module. This "evidence oracle" then returns a yes/no answer to the proposition analyzer to assist in the final interpretation of where the proposition fits with rest of the argument.

This paper focuses on a description of coherent transmission strategies used to guide the proposition analyzer and discusses briefly the necessary working definition of evidence for the evidence verifier to achieve interpretation within the framework of pragmatic analysis. (The role of clues is factored out of the discussion in this paper, but is developed fully in [Cohen 83]).

3. Coherent transmission and reception strategies

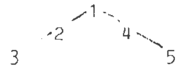
The issues surrounding the operation of the proposition analyzer are discussed in [Cohen 81]. The proposed restrictions to processing are reviewed here, and the implications for a theory of coherence highlighted.

Consider a representation for an argument which is a tree of claim and evidence relations. The root holds the overall point and any son acts as evidence for its father.

One example of a coherent transmission strategy used is PRE-ORDER - the speaker consistently states a claim and then presents evidence for it. A sample argument of this form is presented below:

EX3:
1) Jones would make a good president
2) He has lots of experience
3) He's been on the city board 10 years
4) And he's honest
5) He refused bribes while on the force

with the argument representation:

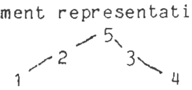


In order for the hearer to reconstruct a pre-order argument he must simply look for a father for each new proposition (NEW). This is done by first testing NEW as evidence for the last proposition (LAST), and if the evidence verification test fails, then the father of LAST and so on, up the right border of the tree. A reception algorithm for pre-order can thus be defined which will successfully assign an interpretation for all propositions in linear time - i.e. the number of operations required to build the overall representation will be proportional to the number of nodes in the tree.

Another coherent transmission strategy is POST-ORDER, where the speaker presents evidence and then states the claim. A post-order form of the argument in EX3 could be as follows:

EX4: 1) Jones has been on the city board 10 years
2) He has lots of experience
3) And he's refused bribes
4) So he's honest
5) He would really make a good president

with the argument representation:

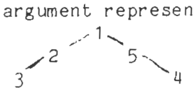


It is possible to describe a reception algorithm for post-order transmission, essentially using a stack to hold sons until the father arrives and reconstructing a sub-tree with that father as root, until the entire tree is built. Once more, the search can be shown to operate in linear time.

As a first approximation to a general processing strategy for the hearer a HYBRID of pre and post order is expected. Now, each particular sub-argument may be transmitted in either pre or post order, as in EX5 below:

EX5: 1) Jones would make a good president
2) He has lots of experience
3) He's been on the board 10 years
4) And he's refused bribes
5) So he's honest

with the argument representation:



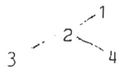
The reception algorithm must now search both for possible father and sons of each new proposition, essentially using a combination of techniques from pre and post order reception algorithms. Considering evidence as a transitive relation - i.e. if A is evidence for B and B is evidence for C, then A is also evidence for C - there is the additional problem that the final representation should reflect the closest evidence relations - i.e. if A is evidence for B and B is evidence for C the tree should simply indicate that A is a son of B, and B is a son of C. Thus, it is possible that a transmission will require re-location of sons in the representation - for example, if the A,B and C discussed above were transmitted in the order: C, A and then B. Then, A would attach to C first and eventually be recorded as evidence for B. Once more, the resulting reception algorithm for HYBRID transmissions can be shown to work in linear time, since the number of possible re-attachments is still limited.

In sum, the proposition analyzer can be restricted to a specified search for possible relatives to the current proposition which performs with reasonable computational effort (linear time). Not all transmissions which can be theoretically generated can be recognized, but the subset accepted is intended to cover coherent structures from the speaker. The theory has been tested on a number of naturally occuring examples from rhetoric books and newspapers (see [Cohen 83]). The transmission forms of these examples conformed to the prescribed restrictions and reasonable representations of the input could be constructed according to the proposed hybrid algorithm. (Hand simulations were done, in the absence of implementation). Moreover, it can be argued that in the absence of any direct indication of structure (linguistic clue) the hearer will not be able to reconstruct the structure underlying transmissions violating these restrictions; he will not expect this more complex strategy.

Note some important characteristics of the transmission forms recognized by the proposed restricted analysis.
1) Not all prior propositions are eligible to receive evidence from the current proposition. Some are closed off from consideration. EX2 shown previously illustrates this property. Here, proposition 3) is no longer a possible father for 5), according to the restrictions of the hybrid reception algorithm.
2) The tests for possible relatives to the current proposition are ordered, so that a relation to a proposition closer to the current one in the utterance stream is found first. Any relation to a proposition further back will not be tested. For example:

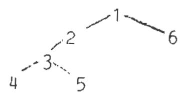
EX6: 1) The city is a mess
2) The parks are a mess
3) The grassy areas are all rotting
4) The gravel paths are run down

Here, 4) is recorded as evidence for 2) rather than for 1), because the connection between 4) and 2) can be established and 2) is closer. (The relation to fill is: "If an area has rotting gravel paths then it is a mess" with the area instantiated here by "parks"). The overall representation looks as follows:



3) It is possible for the current proposition to serve as evidence for a proposition mentioned much earlier in the dialogue. In other words, a relation to the immediate prior statement is not mandatory. For example:
EX7: 1) The city is a mess
2) The parks are a mess
3) The playground area is all run down
4) The sandboxes are dirty
5) The swings are broken
6) The highway system needs revamping

Here, 6) is evidence for 1) because no earlier proposition satisfies a relation to the current proposition. The overall representation is:



Note as well some important similarities between the restrictions for this coherence theory and those required for reference resolution in the work of [Sidner 79] and [Grosz 77]. A set of alternatives is specified. There is a hierarchy of possible choices, with more recent parts of the conversation checked first. The exact components of the hierarchy used to search for the answer to the semantic problem at hand (referent resolution or evidence verification) are specified differently, but the principle is the same: develop a consistent theory for restricting analysis.

4. Evidence verification

The study of evidence relations is a crucial part of the coherence theory of arguments, since the evidence relation is in fact the only coherence relation between propositions in the model. Recall the two main steps to interpreting a proposition according to the model's design: (i) select an eligible prior proposition to test as relating to the current proposition (ii) verify that the evidence relation does hold between the propositions, by approving the necessary missing premises as plausible. The model begins with a shared definition of evidence as follows: a proposition P is said to be evidence for proposition Q if there is some rule of inference such that P is a premise to Q's conclusion. But since analysis is conducted from the hearer's viewpoint, it is possible to elaborate beyond this definition to allow the hearer to recognize evidence relations that use a form of "relaxed logic" or to accept beliefs that he does not currently hold.

It is important to note that the evidence relationship is an underspecified representation tool. The motivation for its use is as follows. In designing an argument understanding system, one fundamental operation for the hearer is to determine the function of each proposition - that is, to which of the other argument propositions it is intended to furnish support. A representation which indicates how all the propositions connect can serve as a model of the plan of the speaker, with the translation: the goal of convincing the hearer of a certain claim is achieved by convincing him that the evidence propositions should be believed. This interpretation of evidence integrates well with Hobbs' description of coherence relations as those which not only link propositions but relate to the goals of the speaker in overall communication ([Hobbs 78]).

The exact details of how the support is to be realized and accepted ultimately by the hearer is not recorded in our analysis model. This falls into the category of judging credibility. However, the evidence relation is intended to cover a variety of possible relations that the speaker could be advocating, including leading the hearer to accept the claim by virtue of a long case analysis or induction.

Consider the case of reasoning by example. The evidence relation would then be understood by the hearer as a modus ponens rule of inference of the form: "If the examples pro are sufficient, then the generalization holds". Of course, with several examples the hearer's credibility will increase. But even with a lone example as support, the hearer can recognize the intended use of the case to support the general claim, especially in the presence of a clue word such as "for example". (More detail on the interpretation of claim and evidence between examples and generalizations, using mutual belief filters, is outlined in [Cohen 83]). This gives only a brief insight into the use of evidence relations to achieve connections of a kind of relaxed logic, in that the inferences contained are not shared "axioms" but defeasible connections.

The effort in defining evidence for this research can be compared to other work in coherence relations. Consider the theories of Hobbs ([Hobbs 76], [Hobbs 78]), employing a set of coherence relations more extensive than claim and evidence, including elaboration, specification, parallel constructions, etc. For our coherence relations we have done the following:
1) provided restrictions on the overall combination of coherence relations within a paragraph of text, by specifying which propositions can possibly be related. This provides a framework for judging coherence of a set of propositions in terms of the coherence relations of the model. [Hobbs 78] does not specify the possible overall combination of relations. In [Hobbs 76] the problem is addressed only partially by specifying a goal list of propositions which have a higher priority to be "related to" by upcoming propositions.

2) provided a general definitional framework for "evidence". Hobbs also provides definitions for his coherence relations in [Hobbs 76]. Our pragmatic approach to analysis, however, provides a framework for recognition of relations beyond that of a shared lexicon and encyclopaedia. We have already shown examples where the hearer may wish to reason beyond his current set of beliefs (EX1). The spectrum of possible tests for evidence verification in fact includes: a) relaxation of logic b) stereotyping the speaker and c) considering what a hypothetical person could be advocating. EX1 illustrates the hearer's attempt at technique c). Below are examples to illustrate variations a) and b).

EX8: 1) Bilandic will win
2) He's the machine candidate

EX9: 1) Reagan is great
2) He stands for apple pie and Mom

In EX8 (drawn from [Sadock 77]), the missing major premise is: "All machine candidates win". But surely, there are exceptions to this statement in real life. The more appropriate belief is: "Most machine candidates win". Thus, the rule of inference connecting 2) to 1) as evidence has a relaxed generalized quantifier.

In EX9, 2) can be evidence for 1) if the hearer believes "If a person stands for apple pie and Mom, then he is great". The hearer may not believe this statement, but still record the evidence relation between 2) and 1) as intended by the speaker if he knows that the speaker is heavily rightwing.

These examples are a necessarily brief discussion of the problem of verifying evidence, but at least emphasize the importance of detailed definitions for evidence within argument understanding systems. (More details of the issues appear in [Cohen 83]).

5. Summary

The theory of coherence for arguments presented here to govern the design of a computational model for argument analysis provides the following framework: transmission forms adhering to the specifications of the hybrid algorithm are coherent; transmission forms beyond these specifications are either a) incoherent or b) coherent in conjunction with the use of additional clues to the structure (e.g. linguistic constructions). Note as well there is extensive variability allowed in the form of acceptable arguments, and in the specification of the evidence relation to be recognized by the hearer.

The main advantage of this research in argument understanding is to move beyond the concerns of other researchers to the fundamental question of recognizing the structural relations of an argument, in order to construct a representation which can be used in subsequent response. We have developed a notion of coherent transmission forms, both to limit analysis to a computationally efficient process and to cover the variety of acceptable input forms for arguments. The coherence theory employs as well a liberal notion of evidence to deal with beliefs not currently held which could still be part of a coherent argument transmission. Once more, this surpasses current research in the field to include necessary pragmatic analyses.

References

[Birnbaum et al. 80] Birnbaum, L., Flowers, M. and R. McGuire; "Towards an AI Model of Argumentation"; Proceedings of AAAI 80

[Birnbaum 82] Birnbaum, L.; "Argument Molecules: A Functional Representation of Argument Structure"; Proceedings of AAAI 82

[Cohen 80] Cohen, R.; "Understanding Arguments"; Proceedings of CSCSI 80

[Cohen 81] Cohen, R.; "Investigation of Processing Strategies for the Structural Analysis of Arguments"; Proceedings of ACL 81

[Cohen 83] Cohen, R.; *A Computational Model for the Analysis of Arguments*; University of Toronto Department of Computer Science Ph.D. thesis

[Grosz 77] Grosz, B.; "The Representation and Use of Focus in Dialogue Understanding"; SRI Technical Note No. 151

[Hobbs 76] Hobbs, J.; "A Computational Approach to Discourse Analysis"; Department Computer Sciences, CUNY Research Report No. 76-2

[Hobbs 78] Hobbs, J.; "Why is Discourse Coherent?"; SRI Technical Note No. 176

[Reichman 81] Reichman, R.; "Plain Speaking: A Theory and Grammar of Spontaneous Discourse"; BBN Report No. 4681

[Sadock 77] Sadock, J.; "Modus Brevis: The Truncated Argument"; in *Papers from the 13th Regional Meeting, Chicago Linguistic Society*

[Sidner 79] Sidner, C.; "Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse"; MIT AI Lab Report TR-537

SCALAR IMPLICATURE AND INDIRECT RESPONSES TO YES/NO QUESTIONS

Julia Hirschberg
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

Abstract

Indirect responses to yes/no questions have commonly been accounted for in terms of the particular "higher goals" of speaker and hearer. However, a form of conversational implicature identified by Horn [8], ***Scalar Implicature***, suggests a more general interpretation. When a cooperative speaker affirms a value on some scale, that value will represent the highest value on its scale the speaker can truthfully affirm. Thus higher values are implicitly marked as false or unknown to the speaker. This paper proposes major extensions to this concept based upon an examination of naturally-occurring question-answer exchanges. It proposes a redefinition of 'scale' and a formalization of the extension for use in computer-human question-answering systems. It describes how the potential scalar implicatures licensed by direct and indirect responses to a yes/no question may be calculated and used to guide the formulation of minimal cooperative responses.

Introduction

A major challenge in natural language processing is how to capture the fact that, in natural discourse, speakers succeed in communicating indirectly as well as directly. Hearers derive more from an utterance than its syntax and semantics convey. In particular, they must often determine not only those propositions that speakers commit themselves to, but also those they do not, to interpret an utterance fully. Speakers in turn take this fact into account when formulating their utterances.

Indirect responses to ***yes/no questions***[a], for example, often permit the inference of both a direct response and additional implicit information. In (1), a questioner (Q) may infer that a respondent (R)'s

(1) Q: Has Jones taken all bis medication?
R: He's had some of it.

direct response to her query is either *no* or *I don't know*. She is also entitled to infer either that R believes Jones has **not** taken *some* of his medicine or that R cannot rule out this possibility. Yet neither of these conclusions can be accounted for by standard models of formal reasoning.

In a first-order representation, R's response in (1), $\exists x(medication(Jones,x) \wedge taken(Jones,x))$, does **not** imply the query's negation, $\neg\forall x(medication(Jones,x) \rightarrow taken(Jones,x))$, or the equivalent inference $\exists x(medication(Jones,x) \wedge \neg taken(Jones,x))$. After all, if Jones has had *some* of his medication it may also be true that he has had *all* of it.

Scholars have long recognized this particular divergence between formal and informal reasoning. However, to date, they have generally explained "non-logical" inferences in exchanges like (1) in terms of particular higher goals of speaker and hearer [7]. For example, if Q and R are medical personnel in (1), they may both realize that Q's higher goal is to make sure Jones has taken his medication but **not** to give him medication he has already taken. So, R's response is appropriate insofar as it furthers this goal.

However, linguistic pragmatics offers a more general explanation of how such inferences may be calculated. A form of ***conversational implicature*** termed by linguists ***scalar implicature*** (SI) provides the basis for an account of exchanges like (1) that is less dependent upon the particulars of context and conversationalists. Observation of naturally occurring dialogues suggest major extensions to this concept.[b] A formalization of an extended notion of SI provides a powerful tool for use in the generation and interpretation of cooperative responses in computer-based question-answering systems.

Scalar Implicature

Grice's [1967] [3] Cooperative Principle postulates that, without contrary evidence, participants in conversation assume their partners are trying to be cooperative. Cooperative speakers recognize certain ***conversational maxims***, which they may use to convey ***conversational implicatures***: A speaker ***conversationally implicates***[c] a proposition P when he conveys the ***implicatum*** P by virtue of the belief he shares with his hearer that CP represents the norm and that the implicatum can be worked out by the hearer. Conversational implicatures may be ***cancelled*** by the linguistic or discourse context: While a speaker who utters 'Some people left early and, in fact, everyone did' ***potentially***[d] implicates 'not all people left early', this potential implicature is intrasententially cancelled. Speakers may not always obey the Gricean maxims: they may ***violate*** them, thus misleading their hearers; they may ***opt out*** of them, making it plain that they are **not** being cooperative; they may be faced with a ***clash*** between two or more maxims and be unable to fulfill all of them; or they may ***flout*** them, causing hearers to search for additional information to explain speakers' deviation from the norm. However, these very notions rely upon speaker and hearer's mutual recognition that CP defines normal conversational behavior.

Of Grice's four original maxims, his ***Maxim of Quality***,

> Try to make your contribution one that is true.
> a) Do not say what you believe to be false.
> b) Do not say that for which you lack adequate evidence.

and his ***Maxim of Quantity***,

> a) Make your contribution as informative as is required (for the current purposes of the exchange).
> b) Do not make your contribution more informative than is required.

provide the basis for a definition of SI.

Horn [8] observed that, if these maxims hold, then when a speaker

[a] Indirect responses to yes/no questions will be defined as responses that fail to provide a direct response of *yes* or *no* or an explicit statement of the respondent's ignorance, e.g. *I don't know*.

[b] The bulk of data examined were transcripts of a radio call-in program, "The Harry Gross Show: Speaking of Your Money", described in [14]. Other data examined are described in [17, 16] or were gathered by Martha Pollack, Ethel Schuster, Gregory Ward, Bonnie Webber, and the author from independent observations. In the interest of brevity and symmetry, the examples presented below are artificial.

[c] *Implicate* is used instead of *imply* to distinguish implicature from logical implication.

[d] Potential implicatures become **actual** implicatures if they are not so cancelled. [2]

refers to a ***scalar***[e] value on a scale defined by ***semantic entailment***[f], that value represents the highest value on its scale the speaker can truthfully affirm. The speaker is saying as much (Quantity) as he truthfully (Quality) can. Higher values on the scale, i.e., those that entail the asserted value, are implicitly marked as either not ***known*** to be true or known ***not*** to be true. Lower values, i.e., those entailed by the asserted value, are true by definition. Horn identifies certain ***scalar predicates***, including logical quantifiers and some connectives, modals, cardinals, ordinals, and miscellaneous modifiers, that support these implicatures. Gazdar [2] later noted that these scalar predicates in effect rank sentences that differ from one another only in a mentioned scalar via this mentioned scalar.

More formally, one might say that, when a speaker R affirms a proposition $\mathbf{P}_{\mathbf{b2}}$ that refers to a value **b2** on a scale **S** defined by semantic entailment, for all propositions $\mathbf{P}_{\mathbf{b3/b2}}$ formed by substituting in $\mathbf{P}_{\mathbf{b2}}$ some **b3** that is higher on **S** than **b2**, R implicates that either he knows $\mathbf{P}_{\mathbf{b3/b2}}$ to be false, $\mathbf{K}_R(\neg\mathbf{P}_{\mathbf{b3/b2}})$,[g] or he does not know whether $\mathbf{P}_{\mathbf{b3/b2}}$ is true or false, $\mathbf{W}_R(\mathbf{P}_{\mathbf{b3/b2}})$.[h] The entire disjunction, $(\mathbf{W}_R(\mathbf{P}_{\mathbf{b3/b2}}) \vee \mathbf{K}_R(\neg\mathbf{P}_{\mathbf{b3/b2}}))$, will be represented below by a **U** operator, where $\mathbf{U}_R(x)$ is equivalent to $(\mathbf{W}_R(x) \vee \mathbf{K}_R(x))$.[i] So, by asserting $\mathbf{P}_{\mathbf{b2}}$, R implicates $\mathbf{U}_R(\neg\mathbf{P}_{\mathbf{b3/b2}})$. For all propositions $\mathbf{P}_{\mathbf{b1/b2}}$ formed by substituting some **b1** lower on **S** in $\mathbf{P}_{\mathbf{b2}}$, $\mathbf{K}_R(\mathbf{P}_{\mathbf{b1/b2}})$ is entailed.

The concept of SI might be applied to question-answer exchanges such as (1) as follows: Q may understand that *some* is the highest value on a quantifier scale *(none/ some/ all)* that R can truthfully affirm. By affirming $\mathbf{P}_{\mathbf{some}}$, R indicates $\mathbf{U}_R\neg$(Jones took all his medication), that is, that either R knows Jones did ***not*** take all his medication or R does not ***know whether*** Jones took all his medication. Thus the direct response to queried *all* is *no* or *I don't know*. Note that, if R simply denied the queried proposition $\mathbf{P}_{\mathbf{all}}$, Q would be entitled to assume that (if R were being cooperative) there was no proposition, say, $\mathbf{P}_{\mathbf{some}}$, that R could affirm. That is, if a cooperative speaker affirms the highest scalar he can truthfully affirm, then a hearer should be entitled to conclude that failure to affirm such a scalar means that the speaker is unable to do so.

Extending Scalar Implicature

While Horn's concept of SI provides a principled explanation for (1), in its current form it is insufficient to account for other exchanges that seem intuitively similar. These exchanges may differ from (1) in three ways: First, they may invoke scales that are neither linear nor defined by entailment. Second, they may include responses affirming a higher value or a value of ***equal rank***, instead of a lower value. Third, they may contain a response that denies or asserts ignorance of some scalar instead of affirming it.

Scales Supporting SI

Many utterances licensing similar implicatures refer to relationships between entities, attributes, events, or states that cannot be adequately represented as linear orderings defined by entailment. Such items may be hierarchically or linearly ordered, as by some whole/part, type/subtype, entity/attribute, set inclusion, temporal, stages of a process, or ordinal ranking, among others. Some of these orderings ***can*** be defined by entailment. In (2), *5 cc. of insulin* might be seen as a

(2) Q: Has Jones had his medication?
R: He's had 5 cc. of insulin.

part entailed by the whole, *Jones's medication*. By affirming *5 cc. of insulin*, R affirms the largest part of the whole he truthfully can. The queried value *Jones' medication* is thus marked by R as unknown or false, $U_R\neg$(Jones has had his medication).

It is less clear that stages of a process may also be defined by entailment, although process metrics support similar implicatures. In (3), *registration* and *filling out insurance forms* may be viewed as

(3) Q: Has Jones registered yet?
R: He's filled out the insurance forms.

stages in a hospital admissions process. By affirming the stage *filling out insurance forms* to a query of the later stage *registration*, R affirms the ***furthest*** stage in the process he can truthfully affirm. Later stages, in particular the queried stage, *registration*, are implicitly marked by R as $U_R\neg$(later stages), while prior stages, say, *filling out a family medical history*, seem to be implicitly affirmed, although they are not entailed.

And some relationships simply cannot be modeled as entailment

(4) Q: Did the chief surgeon concur in the diagnosis?
R: The resident did.

orderings. In (4), for example, the chief surgeon's concurrence clearly does ***not*** entail the resident's. Horn himself notes that the implicatures licensed by reference to scales such as *cold/cool/warm/hot* and *ugly /pretty/beautiful*, and *tort/misdemeanor/felony/capital crime* cannot be accounted for by an entailment definition of scale, although he supplies only ***ad hoc*** solutions. A more general definition of scale seems necessary to capture the full power of SI.

References to Higher or Equal Rank Scalars

Not only is entailment too limited to define the scales that permit SI, but SI conventions may influence the decision to assert a higher scalar than a queried value, and SI also may be conveyed by the assertion of scalars of equal rank with some queried value. In (5), *abdominal pain*

(5) Q: Is Jones experiencing much discomfort?
R: He's complaining of abdominal pain.

may be seen as a subtype of (and thus a higher value[j]) than *discomfort*. R's affirmation of *abdominal pain* allows Q to interpret the direct response as *yes* or *no*, depending upon whether she indeed defines the scale as R does. Similarly, if *having been to X-ray* is a prior stage to *getting results* in (6), then by simple deduction Q can conclude

(6) Q: Has Jones been to X-ray yet?
R: I've got the results right here.

that, by affirming the higher scalar, R has affirmed the lower. Although SI does ***not*** determine Q's interpretation of the direct response in (5) and (6), it ***does*** figure importantly in R's decision not to provide a simple (and truthful) *yes* in either, as will be shown below. And, of course, it does predict Q's inferences about values higher than the asserted scalars, as noted above.

The affirmation of scalars sharing ***equal rank***[k] with a queried scalar may also license implicatures about other values on their scale. In (7), for example, R's assertion of the value *insulin* conveys

(7) Q: Has Jones taken the cortisone?
R: He's taken the insulin.

$U_R\neg$(Jones has taken the cortisone) as well as $U_R\neg$(Jones has taken his medication), if *cortisone* and *insulin* are viewed as parts of that whole, *medication*. That is, R conveys $\mathbf{U}_R\neg$ about other parts of the

[e] A value on some scale

[f] **M** semantically entails **T** iff **T** is true under every assignment of truth values (i.e., in every model) in which **M** is true.

[g] The **K** operator used here is Hintikka's [4] knowledge operator.

[h] This **W** operator is introduced for cosmetic purposes and, in effect, is equivalent to $\neg K_R(x)$, i.e., for R the truth value of x is undefined. Although, in the three-valued logic assumed here, $K_R(x) \vee \neg K_R(x)$ is not tautological -- the tautology is $K_R(x) \vee K_R(\neg x) \vee \neg K_R(x)$ -- **W** is used to prevent confusion over this point.

[i] Below, $U_R(\neg(x))$ will be written $U_R\neg(x)$. Note that $\mathbf{W}_R(\neg x)$ iff $\mathbf{W}_R(x)$.

[j] The truth of a subtype entails the truth of its type.

[k] For the moment this relationship may be likened to that of siblings in some hierarchy.

unmentioned whole, as well as about that whole itself.

Denying and Asserting Ignorance of Scalars

The notion of SI can also be extended to permit the calculation of implicatures licensed by the denial of some scalar or assertion of ignorance of its value. The dual to Horn's original concept would be the negation of the lowest scalar R can truthfully deny. By such a denial R denies higher scalars and affirms or conveys ignorance of lower scalars. So, in question-answer exchanges such as (8) a scalar lower than the queried value may be denied.

(8) Q: Has Jones registered yet?
R: He hasn't signed the release.

If *signing the release* is viewed as a stage prior to registration, then Q can interpret R's response as *no*. By denying *signing the release* R implicates that 'Jones has registered' is also false. For any stage **b1** prior to *signing the release*, R implicates either that he knows **b1** has been completed, $\mathbf{K}_R(\mathbf{b1})$, or that he does not know whether it has been completed, $\mathbf{W}_R(\mathbf{b1})$. This disjunction may thus be represented $\mathbf{U}_R$(Jones has completed **b1**), where $\mathbf{U}_R(P)$ is equivalent to $\mathbf{K}_R(P) \vee \mathbf{W}_R(P)$. Higher scalars may also be denied, as in (9). Q may infer $\mathbf{U}_R$(Jones has been to X-ray), since, in a cooperative exchange, if R could deny the lower scalar, he would do so.

(9) Q: Has Jones been to X-ray yet?
R: I haven't got the results.

So, the denial of $\mathbf{P}_{b2}$ implicates $\mathbf{U}_R(\mathbf{P}_{b1})$ for **b1** lower than **b2**, as, *signing the release* in (8). The value of higher **b3** for scales defined by entailment will be false, by implication.

The notion of SI may also be extended to define implicatures resulting from assertions of ignorance. It follows from Horn's original concept and the extensions to it proposed above that explicit assertions of ignorance of one scalar implicate ignorance of all other values on that scale. A response of *I don't know* may be interpreted as follows: If, in a cooperative exchange, R affirms the highest value on some scale he can truthfully affirm, then his failure to affirm such a value entitles Q to conclude that R can affirm no value on any scale invoked by the query. However, R may also deny the lowest value he can truthfully deny and still produce a cooperative response. Again, his failure to do so entitles Q to conclude that he cannot. So, Q may infer that R can neither affirm nor deny any value on any scale mentioned in the query. That is, $\mathbf{W}_R(\mathbf{bl})$ for any bl on some scalar in the utterance. In (5), for example, *I don't know* would implicate that there is no higher value, i.e., no subtype of type *discomfort*, and no lower value, i.e. no supertype of type *discomfort*, that R can truthfully affirm or deny.

Defining Scales and Formalizing SI

Any ordering that supports SI can be defined as a ***partial ordering***[l] **O** on a collection **B** of referents {**b1,b2,...,bn**}. So, one can view a scale **S** as a *partially-ordered set*. **B** may be finite or infinite, as cardinal scales illustrate. Furthermore, any such poset will support SI, so the poset condition is both necessary and sufficient.

This definition provides a more precise semantics for the informal notions of ***higher***, ***lower***, and ***equal rank*** used above. Given a partial ordering on **B**, for any values **b1**, **b2** on a S, **b2** is ***higher*** on **S** than **b1** iff **b1Ob2**; similarly, **b1** is ***lower*** on **S** than **b2** iff **b1Ob2**. For any pair **b1**, **b2** of ***incomparable elements***[m] of **S** for which there exists a **b3**∈**B** that is ***higher*** than both or that is ***lower*** than both, **b1** and **b2** will be said to have ***equal rank*** on **S**.

So, in (10), on a scale defined by, say, an inclusion relation,

(10) Q: Did you see Catalonia?
R2: I saw Barcelona.
R3: I saw all of Spain.
R4: I saw Valencia.

Barcelona is included in *Catalonia*, so *Barcelona* represents a lower value on this scale in R2; in R3, *all of Spain* is a higher value than *Catalonia*, since it includes that province; and in R4, *Catalonia* and *Valencia* are both included in Spain, but neither includes the other, so they share equal rank.

Given such a definition, implicature may be completely divorced from deduction. An assertion of $\mathbf{P}_{b2}$ may convey for lower values **b1** either $\mathbf{K}_R(\mathbf{P}_{b1})$, or $\mathbf{K}_R\neg(\mathbf{P}_{b1})$, or $\mathbf{W}_R(\mathbf{P}_{b1})$, depending upon the nature of the relation defining the scale, i.e., whether it is defined by entailment or mutual exclusion, or neither. Similarly, a denial of $\mathbf{P}_{b2}$ may ***imply*** for higher values **b3**, $\mathbf{K}_R\neg(\mathbf{P}_{b3})$, or it may convey some other information about these scalars.

With this revised definition of scale and with an extended notion of SI, it is possible to define a set of conventions, $\mathbf{Imp}_{1\text{-}3}$, which, for a given scale, a value on that scale, and an utterance affirming, denying, or asserting ignorance of that value, determine a set of potential implicatures for the utterance. For some scale **S**, a value on that scale **b1**, and a speaker R's utterance **Utt** affirming, denying, or asserting ignorance of a proposition $\mathbf{P}_{b1}$ referring to **b1**, a set of potential implicatures of **Utt**, $\mathbf{PSI}_{Utt}$, may be constructed as follows:

> $\mathbf{Imp}_1$: If **Utt** affirms $\mathbf{P}_{b1}$ then for all **b2** such that **b2** is higher on **S** than **b1**, $\mathbf{U}_R\neg(\mathbf{P}_{b2}) \in \mathbf{PSI}_{Utt}$; and, for all **b3** such that **b3** and **b1** have equal rank on **S**, $\mathbf{U}_R\neg(\mathbf{P}_{b3}) \in \mathbf{PSI}_{Utt}$.
>
> $\mathbf{Imp}_2$: If **Utt** is a denial of $\mathbf{P}_{b1}$ then for all **b2** such that **b2** is lower on **S** than **b1**, $U_R(b2) \in \mathbf{PSI}_{Utt}$; and, for all **b3** such that **b3** and **b1** have equal rank on **S**, $\mathbf{U}_R(\mathbf{P}_{b3}) \in \mathbf{PSI}_{Utt}$.
>
> $\mathbf{Imp}_3$: If **Utt** is an assertion of ignorance of $\mathbf{P}_{b1}$, then for all **b5** on **S**, such that **b5** ≠ **b1**, $\mathbf{W}_R(\mathbf{P}_{b5}) \in \mathbf{PSI}_{Utt}$.[n]

It might well be said that the conventions defined above are, for humans, only conventions, not rules; speakers may ***not*** choose responses that convey as much as they can convey and hearers may ***not*** always expect such responses. However, it is a truism that computer-based question-answering systems should be cooperative. In fact, as Joshi [9] has suggested, users may expect these systems to be even ***more*** cooperative than other humans. Since most such systems must also assume the cooperativeness of their users, it seems reasonable to accept Grice's Cooperative Principle, and, by extension, the SI conventions defined on this basis, as a model for human-machine communication. In the remainder of this paper I will propose some ways in which SI conventions can guide the generation of cooperative responses in question-answering systems.

Providing Cooperative Responses

The notion that question-answering systems should support more 'natural' interaction with their users has long been accepted. This 'naturalness' may in fact involve more than simple 'syntactic sugar'. [10, 14] Studies of human question-answering have found that respondents often provide *more* information than questioners request to anticipate a follow-up question or explain a violated expectation [14]; to correct a misconception [9, 11, 12, 13, 17, 18]; or to satisfy some inferred goal [1, 15]. Joshi's [9] revision of Grice's *Maxim of Quality*: 'Do not say anything which may imply for the hearer something which you the speaker believe to be false.' suggests that respondents often

[l] i.e., a reflexive, antisymmetric, and transitive relation

[m] Elements are incomparable if they are not ordered with respect to one another by **O**.

[n] Note that $\mathbf{W}_R(\mathbf{P}_{b1})$ is asserted, not implicated.

attempt to "square away" any misconceptions apparent in a question or any that they anticipate a questioner might derive from their provision of the information requested. SI conventions provide one account of how respondents may anticipate responses that may be misleading and identify alternatives that are less likely to mislead. That is, using conventions similar to those identified above, respondents may anticipate ***false implicatures*** that might be licensed by a given response. Using the same conventions, respondents may determine ***minimal cooperative responses*** that do not license such implicatures.

Avoiding False Implicatures

The potential false implicatures a given response licenses may be calculated by comparing the implicatures that that response conveys (as defined in $\mathbf{Imp}_1$-$\mathbf{Imp}_3$) with the state of the respondent's knowledge base. Recall that, if speakers are expected to affirm the highest value on a scale they can truthfully affirm, a simple and truthful *yes* to a query of $\mathbf{P}_{\mathbf{b2}}$ will implicate to Q that, for all values **b3** higher on some scale than **b2**, $\mathbf{U}_R\neg(\mathbf{P}_{b3})$. Either $\mathbf{P}_{\mathbf{b3}}$ is false or R does not know its truth value. Hence, if R ***does*** believe some $\mathbf{P}_{\mathbf{b3}}$, then a simple *yes* to the queried $\mathbf{P}_{b2}$ will license a (potential) ***false implicature***, so long of course as Q views **b2** as a scalar and R as cooperative. More generally, for any speaker R, $\mathbf{P}_{\mathbf{b3}}$ is a potential false implicature of $\mathbf{P}_{\mathbf{b2}}$ iff in some context **D**, R's affirmation of $\mathbf{P}_{\mathbf{b2}}$ implicates a truth value for $\mathbf{P}_{b3}$ that is inconsistent with R's knowledge of $\mathbf{P}_{\mathbf{b3}}$. If this implicatum is not ***cancelled***[o] in the discourse, then R should try to formulate an alternative response that will not carry this risk of misleading Q.

So *yes* in (6), for example, might mislead in the following sense: Affirming the value, *going to X-ray*, for Jones, affirms a value that is ***not*** the highest value on its scale, *getting and processing an X-ray*, that R can truthfully affirm. Thus (incorrectly) higher values, such as *getting back the results*, are marked as false or unknown by R. So Q will be entitled to conclude that R does ***not*** know that *getting back the results of an X-ray* is true for Jones -- when in fact he does. Similarly, in (3), a simple *no* also fails to affirm the highest value on the scale *getting into the hospital* R knows to be true. In (2), a direct response of *no* implicates that there is no ***lower*** value, no part(s) of the whole, that R can affirm.

Recall also that R may deny the lowest value on some scale that he can truthfully deny. Such a denial implicates that lower values **b1** on a scale are not known to be false, i.e., are either known to be true or have an unknown truth value. Thus if R knows some **b1** to be false, a simple *no* to a query of a higher **b2** will be true but possibly misleading. For example, in (8), a simple *no* denying the stage *registration* would implicate $\mathbf{U}_R$(Jones has signed the release), when, in fact, R knows Jones has ***not*** completed this stage.

The notion of SI may also be extended to define implicatures resulting from assertions of ignorance. As noted above, explicit assertions of ignorance of one scalar implicate ignorance of all other values on that scale, that is, $\mathbf{W}_R(\mathbf{bl})$ for any scalar **bl**. If, in fact, R ***can*** affirm or deny some such value, then an expression of ignorance will license false implicatures about other values on any scale invoked in the query. In (5), for example, *I don't know* would implicate that there is no higher value, i.e., no subtype of type *discomfort* that R can truthfully affirm or deny. Thus Q is entitled to assume not only that R is ignorant of whether or not Jones suffers from any discomfort, as the response entails, but also that R is ignorant of whether Jones suffers from any subtype of *discomfort*, such as *abdominal pain*, or any supertype of *discomfort*, say, *general complaints*.

So the concept of SI can help determine that an indirect response is more appropriate than a direct response when the latter would entitle Q to derive false implicatures. But even if R anticipates such implicatures, why might he choose to avoid them by making an indirect response instead of, perhaps, a qualified direct response?

Minimal Cooperative Responses

The appropriateness of the indirect responses in the exchanges presented above depends in part upon the relative inappropriateness of alternative responses. Suppose that possible (truthful) direct responses to yes/no questions could be placed on a continuum according to amount of information explicitly provided, from a simple direct response through various ***modified direct responses***[p], to a ***complete and unambiguous response*** (CUR). A CUR will be defined as a response in which R attempts, in some context **D**, explicitly to ***square away*** [9], or resolve, all misconceptions which might arise in **D** if he provides only the information requested by Q.

We have seen above that simple direct responses may mislead by licensing false implicatures. Suppose instead that R attempts to provide a CUR to the query in (2), as in (11):

(11) Q: Has Jones had his medication?
R: No. He's had 5 cc. of insulin, which is part of that
medication, but he hasn't had the rest of that medication,
that is, the liprin, the cortisone.... So, he hasn't had his
medication.

This alternative, like that in (2), certainly anticipates and avoids potential false implicatures. But verbosity obviously lessens the appeal of such responses: Q may be overwhelmed by unrequested information which R must laboriously gather and present, and which may be redundant or irrelevant for Q.

Grice notes that speakers generally prefer to communicate via the briefest utterance that conveys what they wish to convey. By taking advantage of the SI conventions, R may limit the explicit information he must produce in his response in several ways: First, he may omit the unnecessary simple direct response, since, as we have seen, the hearer can infer it. Second, he may omit an explicit statement of additional unrequested information that is implicated by his response, avoiding some risk of irrelevance and redundancy by ***permitting*** inference of propositions instead of asserting those propositions. So, SI conventions provide a principled way of limiting the amount of information explicitly provided in a response. As noted above, SI conventions may also help determine when responses will license false implicatures, and, by extension, which responses will not license false implicatures. So, in terms of the potential a response may have for misleading a questioner as well as in terms of the amount of explicit information provided in a response, SI conventions provide one method of determining ***minimal cooperative responses*** in question-answering systems.

With a definition of scale, conventions for identifying potential SI, and some understanding of how these conventions might define a minimal cooperative response, it is possible to describe the process of identifying such a response in general terms:

1. Q queries some proposition $\mathbf{P}^{(q)}$ in a discourse context **D**;

2. For each **bl** referenced in **P** ($\mathbf{P}_{\mathbf{bi}}$) that R recognizes as lying on some scale $\mathbf{S}_i$

   a. $\mathbf{P}_\mathbf{x}$ represents the open proposition formed by substituting a variable **x** for **bl** in $\mathbf{P}_{\mathbf{bi}}$;

   b. R instantiates $\mathbf{P}_\mathbf{x}$ with some **bj** that co-occurs with **bl** on $\mathbf{S}_i$ and determines the truth-value of the resulting proposition $\mathbf{P}_{\mathbf{bj}}$;

   c. R determines that $\mathbf{P}_{\mathbf{bj}}$ has not been asserted in **D**;

   d. Using $\mathbf{Imp}_1$-$\mathbf{Imp}_3$, R calculates the set of implicatures $\mathbf{PSI}_{\mathrm{Utt}}$ licensed by asserting the truth-value of $\mathbf{P}_{\mathbf{bj}}$;

[o] See [2, 6] for a discussion of cancellation.

[p] Simple direct responses with some additional information

[q] That is, P represents that query's **desideratum** [5], 'That which the questioner desires to be made known to him'.

e. R examines $\mathbf{PSI}_{\mathbf{Utt}}$ to determine whether these implicatures are consistent with his knowledge base or whether any that are not will be cancelled in **D**;

f. If $\mathbf{P}_{\mathbf{bj}}$ has not been asserted, and if it licenses no false uncancelled implicatures in **D**, R will realize $\mathbf{P}_{\mathbf{bj}}$ in **Utt**.

The following example illustrates this model:

(12) Q: Is the Pacific Fleet in port?
R: The First Battle Division is.

The proposition 'the Pacific Fleet is in port' represents the desideratum of Q's query. R perceives *Pacific Fleet* as lying on a whole/part scale **F** defined by inclusion. The open proposition $\mathbf{P}_{\mathbf{x}}$ formed from $\mathbf{P}_{\text{Pacific Fleet}}$ is '**x** is in port'. R instantiates $\mathbf{P}_{\mathbf{x}}$ with *the First Battle Division*, which occurs on **F** with *Pacific Fleet*, producing 'the First Battle Division is in port' ($\mathbf{P}_{\mathbf{bj}}$). From his knowledge base R calculates the truth-value of $\mathbf{P}_{\mathbf{bj}}$, and finds it ***true***. He determines from **D** that 'the First Battle Division is in port' has not been asserted in the discourse. R then calculates the potential implicatures licensed by an assertion of 'The First Battle Division is.', given **F**. Using $\mathbf{Imp}_1$ R determines that this assertion would license the potential implicatures $\mathbf{U}_R\neg$(the Pacific Fleet is in port, the Second Battle Division is in port,...). R finds these implicatures consistent with his knowledge base. If no other scalars pass the tests for inclusion in his response, R will decide that 'The First Battle Division is.' represents a minimal cooperative response to Q's query.

The discourse model sketched above is somewhat oversimplified. Questions of when possible scalars are actually viewed as such, how Q and R recognize a common scale for any such scalar, and how multiple scalars are accommodated in a single response add to the model's complexity and are now under investigation. However, it should serve to illustrate the way a formalization of SI can serve to guide the generation of minimal cooperative responses to yes/no questions in computer-based question-answering systems. A module of a natural language interface to such a system that uses a more complex version of this model to provide such responses is currently being developed.

Conclusion

In this paper I have proposed an extension of the notion of SI, defining scale more generally to encompass additional metrics, identifying implicatures licensed by denials and assertions of ignorance, and recognizing the parallels between references to lower and higher scalars and scalars of equal rank. I have presented a simple formalism of this extended notion as well as a general model for its use in answering yes-no questions. Together these permit a principled account of

1. how respondents might decide a simple direct response is misleading,

2. how respondents might choose a minimal cooperative response that is not, and

3. how questioners might derive a simple response and additional inferences from that response.

Although I have focussed here upon the possibilities SI presents for the ***generation*** of cooperative responses, the same conventions should be pertinent to natural language ***understanding*** as well. In particular, the process suggested above for identifying potential false implicatures might be turned to the task of recognizing implicatures licensed by user assertions, again permitting more natural interaction between system and user. Thus SI might make the extraction of information from users more efficient as well as making the responses provided to those users more cooperative.

Acknowledgements

This work has been partially supported by an IBM Graduate Fellowship. I would like to thank Tim Finin, Aravind Joshi, Kathy McCoy, Kathy McKeown, Martha Palmer, Martha Pollack, Ellen Prince, Ethel Schuster, Gregory Ward, and Bonnie Webber for their help on this paper and earlier versions.

References

[1] Allen, James F. and C. Raymond Perrault. "Analyzing Intention in Utterances." *Artificial Intelligence 15* (1980), 143-178.

[2] Gazdar, G. A Solution to the Projection Problem. In *Syntax and Semantics*, Oh, C.-K. and Dinneen, D., Eds., Academic Press, New York, 1979.

[3] Grice, H. P. Logic and Conversation. In *Syntax and Semantics*, Cole, P. and Morgan, J.L., Eds., Academic Press, New York, 1975.

[4] Hintikka, J.. *Knowledge and Belief*. Cornell University Press, Ithaca, 1968.

[5] Hintikka, J. Answers to Questions. In *Questions*, Hiz, H., Ed.,Reidel, Dordrecht (Neth.), 1978, pp. 279-300.

[6] Hirschberg, J. Scalar Implicature and Indirect Responses to Yes-No Questions. Tech. Rept. MS-CIS-84-9, University of Pennsylvania, April, 1984.

[7] Hobbs, J. and Robinson, J. "Why Ask?" *Discourse Processes 2* (1979).

[8] Horn, L. R. *On the Semantic Properties of Logical Operators in English*. Ph.D. Th., University of California at Los Angeles, 1972.

[9] Joshi, A.K. The Role of Mutual Beliefs in Question-Answer Systems. In Smith, N., Ed., *Mutual Belief*, Academic Press, New York, 1982.

[10] Joshi, A.K, and Webber, B.L. Next Steps in Natural Language Interaction: Beyond Syntactic Sugar. Proceedings of the Conference, Fourth Jerusalem Conference on Information Technology, May, 1984. Forthcoming.

[11] Kaplan, S.J. Appropriate Responses to Inappropriate Questions. In *Elements of Discourse Understanding*, Joshi, A.K., Webber, B.L., and Sag, I.A., Eds., Cambridge University Press, Cambridge, 1981, pp. 127-144.

[12] Mays, Eric. Correcting Misconceptions about Data Base Structure. Proceedings of the Conference, Canadian Society for Computational Studies of Intelligence, 1980.

[13] McCoy, K. Cooperative Responses to Object-Related Misconceptions. Tech. Rept. MS-CIS-83-89, University of Pennsylvania, November, 1983.

[14] Pollack, M. E., Hirschberg, J., and Webber, B. User Participation in the Reasoning Processes of Expert Systems. Tech. Rept. MS-CIS-82-9, University of Pennsylvania, July, 1982. A shorter version appears in the AAAI Proceedings, 1982

[15] Pollack, M.E. Generating Expert Answers through Goal Inference. Tech. Rept. 316, SRI International, October, 1983.

[16] Prince, E.F., Frader, J. and Bosk, C. On Hedging in Physician-Physician Discourse. In *Linguistics and the Professions*, De Pietro, R., Ed.,Ablex, Norwood (N.J.), 1982, pp. 83-97.

[17] Schuster, E. Explaining and Expounding. Tech. Rept. MS-CIS-82-49, Computer & Information Science, Univ. of Pennsylvania, 1982.

[18] Webber, B.L., and Finin, T. In Response: Next Steps in Natural Language Interaction. In *Artificial Intelligence Applications for Business*, Reitman, W., Ed.,Ablex , Norwood, N.J., 1984.

Generating Corrective Answers by Computing Presuppositions of Answers, Not of Questions or Mind *Your* P's, not Q's[1]

R.E. Mercer and R.S. Rosenberg
Department of Computer Science
University of British Columbia

Abstract

The standard approach for an Answerer to generate *corrective* as opposed to *direct* answers to a question is to compute the "presupposition of the question", interpret this statement as a belief of the Questioner, and, if this belief is false, to correct the false belief. We suggest a new approach based on computing the presupposition of the answer, for the following reasons: (1) one does not need to compute the "presupposition of the question" to be able to *justify* and *form* a corrective answer, since one can accomplish these ends by computing the presupposition of the answer, (2) the new approach is successful in a class of Question-Answer situations for which the standard approach was not designed, and (3) the new approach generates the desired direct and corrective answers within a well established theory (Gazdar, 1979).

Introduction

The Question-Answer situation is a planning process in which the Questioner (**Q**) plans questions according to some desired outcome and the Answerer (**A**) plans replies according to the truth value of the available information. Yes/no, wh-, and how many-questions concern us here. The following are examples of these types of questions. In each of the the three examples Q is the question, P is the "presupposition of the question"[2], A1 represents a *positive direct answer* that entails P, A2 a *negative direct answer* that presupposes P, and A3 is a *corrective answer* which denies P. The answer can be shortened by deleting the restatement of the (modified) question. The meaning of the shortened answer, shown in parentheses, is taken to be that of the full answer.

Q: Have you stopped beating the rug?
P: You have been beating the rug.
A1: Yes, I have stopped beating the rug. (Yes.)
A2: No, I have not stopped beating the rug. (No.)
A3: No, I have not stopped beating the rug, because I haven't started. (No, because...)

Q: Which boys that went to the circus went to the movies?
P: Some boys went to the circus.
A1: John and Bill, the boys that went to the circus, went to the movies. (John and Bill.)
A2: No boy that went to the circus went to the movies. (None of the boys.)
A3: No boy that went to the circus went to the movies, because no boy went to the circus. (None, because...)

Q: How many boys that went to the circus went to the movies?
P: Some boys went to the circus.
A1: Three boys that went to the circus went to the movies. (Three.)
A2: None of the boys that went to the circus went to the movies. (None.)
A3: None of the boys that went to the circus went to the movies, because none of the boys went to the circus. (None, because...)

The standard approach[3] to the Question-Answer situation, which we call P_Q, computes the "presupposition of the question" and generates answers to correct false presuppositions (Belnap and Steel, 1976; Kaplan, 1982). This method provides **A** with two sources of information to form an answer: **A**'s knowledge of the domain about which **Q** is requesting information and a supposed subset of **Q**'s beliefs which **A** derives as the "presupposition of the question". The "presupposition" is inferred from lexical items and syntactic constructions in the question. The examples above demonstrate this for "stop" and relative clauses. **A** plans answers in the following way: If the presupposition is true then **A** can give a direct answer. If it is false then **A** is required to give a corrective answer which appears to correct the false belief rather than answer the main question.

The alternative approach, presented here, forms answers with presuppositions that can be proved true. This method, here called P_A, produces answers from **A**'s point of view

[1] This research was supported in part by the Natural Sciences and Engineering Research Council of Canada grant A7642 (to R. Reiter).

[2] Scare quotes are used to indicate that we are not committed to the existence of such an entity.

[3] Comments throughout are about the standard approach in general, but since Kaplan(1982) is contained in a computational framework and since it is representative of the standard approach, we will make comparisons with it specifically.

regardless of **Q**'s (supposed) beliefs. It is based on Gazdar(1979)'s interpretation of the Maxim of Quality of the cooperative principle (Grice, 1975); that is, **A** cannot utter an answer that has non-true presuppositions. In addition, Gazdar's theory for computing presuppositions is used by **A** to plan the form of the answer. This theory states that **Q** will infer the presuppositions of the answer from lexical items and syntactic constructions contained in the answer unless the presupposition is inconsistent with the context. The context includes general real world knowledge, specific knowledge that **Q** has, and information contained in the sentence. **A**, applying this theory, plans the answer in the following way: If the presupposition is true then **A** can give a direct answer. If the presupposition is not provably true then since **A** knows that **Q** finds the presupposition assumable (ie does not believe it to be false (Kaplan,1982)), **A** must supply enough extra information (possibly in the form of a because-clause) so that **Q** will not infer the (non-true) presupposition and will interpret the sentence properly. Details and examples are given in the next section.

For questions that ask about objects and relations in closed extensional data bases, P_Q is sound and complete. P_A should be viewed as generalizing the cooperative behaviour found in P_Q to data bases that are not closed, allow inferencing, and are incomplete with respect to the questions that can be asked.

Description of the Two Methods

Given the previous examples we realize that the structure of the answers for yes/no, wh-, and how many-questions are just modifications of the question's syntax, together with appropriate additions such as yes or no for yes/no questions, a list for what-, which-, or who-questions or a number for how many-questions, and because-clauses for corrective answers. Since this is the case, P_Q and P_A can generate an answer from some simple set of rules and sentence schema. We now look in more detail at how the two methods compute answers.

Both methods transform the question into a query in some query language, and present it to a data base. If the answer is positive, both methods produce an A1 answer. If the answer is negative, the two methods take different approaches.

P_Q does the following for negative answers. The "presupposition of the question" is computed. It is then passed to the data base as a query. If the query is true then an A2 answer is formed. If the query is false an A3 answer is created. The content of the because-clause is computed according to an algorithm which finds the smallest failing subgraph in a Meta-Query Language (MQL) graph (Kaplan, 1982). The details are not important, but it essentially finds the most general correction.

P_A works in the following way for negative answers. **A** first forms an A2 answer. Because **A** is committed to the Maxim of Quality, **A** cannot utter the answer unless the presupposition of the answer is true. The presupposition is computed (computation proceeds as in Mercer and Reiter(1982)), translated into a query, and passed to the data base. The presupposition can be proved true, proved false, not provably true or false, or the proof may be terminated early because of resource limitations. We consider each of these outcomes.

(1) If the presupposition is true then the A2 answer can be uttered.

(2) If the presupposition is false then an A3 answer is formed with a because-clause that denies the presupposition. Noting that an A3 answer may have a presupposition, this answer is subjected to the same process again. If the new presupposition is true the A3 answer is uttered. If it is not true then the because-clause is replaced by another because-clause which denies the new presupposition. This cycle continues until the A3 answer has only true presuppositions. We end up generating a sentence with the most general correction. This is analogous to P_Q which finds the smallest failing subgraph in MQL graphs.

(3) If the presupposition cannot be proved true or false, or if the proof is terminated early, then a different kind of answer must be generated -- an answer like "I don't know (if...) because I don't know if 'the presupposition' is true." And like the answers in case (2), this sentence may have new presuppositions that need to undergo the same cycle as in case (2). The result of any of these steps may be an A3 answer or an "I don't know..." answer.

If the answer cannot be proved true or false the two methods are different. P_Q gives as a solution an "I don't know..." answer. But this answer is potentially incorrect because it may have false presuppositions. P_A processes this class of answers in the same manner as negative answers. The following example illustrates the differences between the two approaches.

Question: "Do any professors that teach CPSC101 teach CPSC114?"

which "presupposes"

"There are professors that teach CPSC101.".

The query, in some query language, would be

"Does there exist an individual that teaches CPSC101 and teaches CPSC114?".

(1) Suppose the data base can prove the query false, and can prove that the presupposition is true. Then the answer for both P_Q and P_A would be an A2 answer, that is, "No."

(2) Suppose the data base can prove the query and the presupposition false. Then the answer for both methods would be an A3 answer, that is, "No, because no professor teaches CPSC101.".

(3) Suppose the data base can prove the query false, (say, by proving that no professor exists that teaches CPSC114) but cannot prove that the presupposition is true or false. Because the presupposition cannot be proved false, P_Q must give an A2 answer. This answer is wrong in this case because an A2 answer presupposes (ie **A** is communicating to **Q** that the data base can prove) that "There are professors that teach CPSC101.". P_A can give an A3-like answer, that is, "No, and in addition I don't even know if any professor teaches CPSC101." Note that P_A requires that any new presuppositions be treated like the original one. So the presupposition "CPSC101 is taught." must now be proved. If the data base proves it true (say, by proving that sessional instructors teach CPSC101), the answer is not changed. If the data base proves it false, then the answer must be changed to "No, and in addition CPSC101 is not taught.". If the data base cannot prove it either true or false, then the answer must be changed to "No, and in addition I don't even know if CPSC101 is taught.".

(4) Suppose the data base cannot prove the query true or false. In addition the presupposition cannot be proved true or false. P_Q would answer "I don't know." which would be incorrect because this answer presupposes that there are professors that teach CPSC101. On the other hand, P_A would respond with "I don't know, and I don't even know if any teach CPSC101."

Differences between P_A and P_Q

P_Q works correctly with closed extensional data bases and for questions that can be answered by the data base; that is, questions that ask about objects and relations in the data base. Also it is complete for these types of data bases and questions. P_Q and P_A coincide in this restricted environment. P_A is intended to work additionally with data bases that are not closed, that allow inferencing, and that are incomplete with respect to the questions that can be asked.

The major difference (other than the obvious difference between computing presuppositions from questions or answers) between P_A and P_Q is that where P_Q must prove the presupposition false in order to give a corrective answer, P_A need only fail to prove it true in order to generate a corrective answer.

P_Q takes the "presupposition of the question" as a belief of **Q**. To correct **Q**, **A** must prove that this belief is false. From this point of view, this is a reasonable rule; **A** should not correct a belief of **Q** unless **A** is certain that it is false. Simply changing P_Q to allow answers similar to those generated by P_A is not justified, because correcting **Q** would be allowed even though **A** is not sure that **Q** has false beliefs. A shift in point of view from computing the "presupposition of the question" to computing the presupposition of the answer must be made in order to justify the switch from having to prove false to failing to prove true.

Conclusion

We have presented an alternative to the standard approach for generating corrective answers to questions. We have demonstrated that the new method not only works for those Question-Answer situations for which the standard approach works, but also for the situation in which a proof of the truth or falsity of the question or the presupposition is unavailable. In addition the algorithm uses a well-established theory for computing presuppositions (Gazdar, 1979).

Acknowledgements

We would like to thank Dave Etherington for his comments on an earlier draft of this paper.

References

Belnap, N.D. and T.B. Steel (1976). *The Logic of Questions and Answers*, Yale University Press.

Gazdar, G.J.M. (1979). *Pragmatics Implicature, Presuppositions, and Logical Form*, Academic Press.

Grice, H.P. (1975). Logic and Conversation, in *Syntax and Semantics Vol. 3 Speech Acts*, P. Cole and J.L. Morgan (eds), Academic Press.

Kaplan, S.J. (1982). Cooperative Responses from a Portable Natural Language Query System, *Artificial Intelligence 19.*

Mercer, R.E. and R. Reiter (1982). *The Representation of Presuppositions Using Defaults*, in Proc. 4th CSCSI/SCEIO Conference.

GOOD ANSWERS TO BAD QUESTIONS: GOAL INFERENCE IN EXPERT ADVICE-GIVING

Martha E. Pollack
Department of Computer and Information Science
The Moore School
University of Pennsylvania
Philadelphia, Pa. 19104

Abstract -- In the past expert systems have adopted the *appropriate-query assumption*--that people consulting them always ask for precisely the information they need. In fact, people often need to consult with experts because they have an incomplete understanding of their options and do not know what it is that they need to know. It is a significant feature of human expertise to be able to deduce, from an incomplete or inappropriate query, what advice is actually needed. This paper describes ongoing research to incorporate similar inference capabilities in automated expert systems so that they too will be able to deduce advice-seekers' goals and thereby generate appropriate answers to their queries. A framework for inferring goals is described along with a strategy for controlling the inferences.

Introduction

Imagine that you're trying to use MM, a mail system on the DEC-2060 computer, and that partway through creating a message you accidently type a Control-Z, which has the pernicious effect of ending your message creation and sending you to the prompt level. You turn to your ever-ready expert and ask if there's some way to undo a control character from within the mail system. Your expert tells you that there isn't, and so, feeling annoyed and frustrated, you proceed to recreate your message from scratch. When you later discover that you could indeed have entered an editor from the prompt level and added directly to the existing message fragment, you begin to doubt your expert's expertise.

Now imagine that you've just inherited $120,000. You immediately call an investment counselor to ask whether you'll earn more interest on Treasury-notes or on certificates of deposit. The counselor asks you a few questions about your tax bracket and other investments, and then informs you that T-notes are the better investment. What she doesn't tell you is that, given her knowledge of your financial situation, municipal bonds would be a far better investment than either of your expressed alternatives. When you later learn this fact, you're very likely to find a new investment counselor.

In each of these scenarios, you've gone to an expert seeking advice about some problem, and in each the expert has failed to provide you with the most appropriate advice. The failure resulted from the expert's assumption that you knew exactly what advice you needed, and that you had accurately and literally expressed a request for that advice in your query. This assumption, which I call the *appropriate-query assumption*, has been made in most existing expert systems. Yet observation of dialogues between human experts and advice-seekers reveals that it is much too strong. People often need to consult with experts precisely because they do not know what it is that they need to know: they may have an incomplete notion of what their options are -- or perhaps no good notion at all. As a result they may intend to do something that is not actually the best thing they could do.

It is a significant feature of human expertise to be able to deduce, from an incomplete or inappropriate query, what advice is actually needed. The goal of the ongoing research described in part here is to enable automated experts to perform similar deductions, and thereby generate appropriate answers to queries made to them. This paper describes the components of a framework for goal inference, along with a strategy for controlling the inferences.

Assumptions

Two distinct types of goals can be associated with any query to an expert. Previous work has often been directed to the inference of *communicative* or *illocutionary* goals of indirect speech acts. For example, [4, 9] present a theory relating the query "Is there some way to undo a Control-Z?" to the goal of finding out a procedure for undoing a Control-Z. In contrast, the theory being developed here assumes the inference of the communicative goal and emphasizes the deduction of what I shall call *domain* goals: it relates a request to be told how to undo a Control-Z to an unexpressed request for a method by which to complete a partial mail message.

The inference of domain goals has been studied within the context of one agent observing another's actions [5, 6, 13] or reading about them [2]. This differs from the context assumed in current work, in which the expert must infer the advice-seeker's goal from the discourse rather than from observing his actions. The difference in assumptions is critical to the implementation of this work in systems that are expert on some domain external to the computer: such systems can only receive their information from discussion with the advice-seeker.

Research into domain goal deduction based on discourse [1, 3, 7] has generally made the appropriate-query assumption, that the advice-seeker's query requests information that he actually needs. When the appropriate-query assumption is made, domain goals are deduced in order to provide additional information or to respond to query fragments. The one current system that appears not to be making the appropriate-query assumption is the Consul system [8]. However, it does make the strong assumption that the advice-seeker has a complete model of one domain (postal mail) which is distinct from the system's domain of expertise (electronic mail) and which is specifiable *a priori*. What Consul provides is a mechanism for mapping between the two domains. In fact it does implicitly assume the appropriateness of the advice-seeker's query, but with respect to the former domain.

To summarize, the research described in this paper differs from earlier work in that it studies the inference of domain goals, based only on discourse with an advice-seeker, without making the appropriate-query assumption.

The Goal-Inference Process

As already mentioned, existing expert and planning-systems make the appropriate-query assumption: they adopt the advice-seeker's expressed goal, and attempt to deduce either a fact that he says he wants to know or a plan that achieves a state of affairs he says he wants to obtain. However, as shown in the introduction, a cooperative expert cannot assume that the most appropriate response to a query addresses the expressed goal. Often the best response will address some domain goal unexpressed in the query. This section maps out the steps needed to determine what that goal may be.

The analysis of goal inference described here was motivated by the study of dialogues between human experts and advice-seekers.[a] For

[a] Thanks are due to M. Tyson and J. Hobbs; B. Lewis and E. Goldberg; and E. Eceli, D. Klein, and P. Natali for collecting the transcripts.

the initial study, the dialogues came from several sources, but in all cases the domain of expertise was a computer system such as MM or the text editor EMACS. Computer systems provided an attractive domain for the preliminary analysis because of several simplifications they permit to the general problem of providing an appropriate answer:

- There is only one agent operating, apart from the system. Interactions among multiple agents' goals need not be considered.

- The effects of actions can be assumed to be certain. When an MM user, for example, types "SEND", one can assume that this results in his current message being sent. No such assumptions can be made about the effect, say, of investing money.

- There is a limited but nontrivial number of potential actions the user can perform. The number is small enough to make an axiomatization feasible. However, this small set of actions can be combined and structured in interesting ways.

Allowing Different Query Types

Even a cursory examination of naturally occurring dialogues reveals the first way in which the appropriate-query assumption is overly restrictive. Advice-seekers do not always directly specify the state of affairs they want. The cooperative expert must use what information is provided in the query to infer the advice-seeker's goal.

Study of the transcripts mentioned above has shown that a large percentage of queries are of three types. The first type can be paraphrased as "How can I have condition P hold?" (where P may be some condition or a Boolean combination of conditions); the second as "How can I do action α?" (where α may be some action or a Boolean combination of actions); and the third as "How can I perform some action α but have modifying condition P hold?".[b] Of course, there are queries that do not fall into one of these types; identifying and analyzing other query types is part of the ongoing research.

In the goal-inference framework I am constructing, each query type is associated with a *target predicate*. The target predicates for the query types mentioned above are wantState(A,p), wantAction(A,α), and wantModifiedAction(A,α,p), respectively, where A indexes the advice-seeker and p and α encode the formula(s) and/or action(s) he says he wants. I am assuming the existence of a semantic analyzer to provide logical forms that could then be translated in a principled way into such a set of predicates. The target predicates serve as input to the goal-inference system.

To see how potential goals are inferred from target predicates, let us consider how a human expert deals with queries of each type. To begin consider the second query type, in which the advice-seeker describes in his query an action that he wants to perform. This is the type of the query in the Control-Z example. When presented with the description of an action, the human expert infers that a potential goal of the advice-seeker is the effect of the described action. So, for example, when asked "How do I send a message to Joe?", the expert will provide a plan that achieves the result of the action "send a message to Joe". This may seem somewhat paradoxical. If what the expert is going to provide is a plan that will achieve the advice-seeker's goal, then why does the advice-seeker specify to the expert a plan whose effect will be that goal? That is, if the advice-seeker already knows the action he wants to perform, why does he bother the expert at all? The answer to this is that the advice-seeker does not know *how* to perform the action he is describing: what he conveys to the expert is an action description, not an actual, executable plan.

Sometimes the action described in the query may be extremely close to an executable sequence of actions, particularly in a computer-system domain. This results in exchanges like the following:

A: "How do I delete a message?"

E: "Would you believe DELETE followed by the message number?"

Because the action description "delete a message" happens to be nearly identical to the MM command DELETE, it is easy to confuse the two. However, in another mail system, deleting a message might require that you first read the message and then type "DESTROY," so that the exchange between expert and advice-seeker instead looks like this:

A: "How do I delete a message?"

E: "Type READ n, where n is the number of the message you want to delete, and then type DESTROY."

Although the query contains the same action description as the previous example, the answer consists of a different executable plan.

The following three examples, taken from the transcripts, provide further evidence that experts often take the effect of a described action, rather than the action itself, to be what the advice-seeker wants.

A: "I'm getting some inconsistency here. I wanted to add some addresses to my LETTER file and I used APPEND to do it. But nothing's happening. How can I get APPEND to work?"

E: "Well, LETTER is unsupported software so if it isn't working there isn't much you can do. What you can do is set up a separate, new file and then just use SOS to copy that file to the end of the other one."

Note that here the expert is unable to give the advice-seeker a way to fix the APPEND command. However, she knows what the result of fixing APPEND would be, and she provides a plan that achieves that result.

The next example is similar:

A: "I'm trying to use the Diablo printer, but there's no ribbon in it. Could you put one in for me?"

E: "No. You have to have your own ribbon. You can buy one at the computer store."

Here the expert's answer is in the same spirit as the previous example: although the requested action cannot itself be performed, the results that that action would have if it were performable can be independently planned for. The expert cannot herself put a ribbon into the printer, but she can and does tell the advice-seeker an alternative way to achieve what would be the result of her inserting a ribbon.

Here is a third example, from an EMACS user:

A: "How can I repeat a command? Control-K?"

E: "Most likely what you should do is define a region, and then just kill that region."

The advice-seeker here describes an action he would like to perform: his query would be represented in the framework as wantAction(A,repeat(Control-K)). What the expert does is determine the effect of the action he wants. Control-K deletes a line, so repeated Control-K's will delete several lines. The expert then tells him a plan to achieve that effect. Note that the given plan does not involve repeating Control-K's at all. The expert has produced a plan to achieve the effect of the action α (repeat(Control-K)) independent of α itself.

[b] Another class of questions is quite common. They occur when something has "gone wrong" and the advice-seeker asks either "How did I end up with condition P holding (when I wanted condition Q)?" or "I performed action α, what conditions hold now?" Analysis of this class has been deferred because it requires an additional layer of reasoning.

Linking Rules

The expert's ability to infer a potential goal from the query is captured in the goal-inference framework by a set of *linking rules*. Linking rules link together target predicates and potential goals. The linking rule that applies to the target predicate wantAction(A,α) follows:

Linking Rule 1. If an advice-seeker A describes an action α that he wants (i.e. the target predicate is wantAction(A,α)) and the result of performing α in the current state of affairs would be the state of affairs p, then p is a potential goal of A.

Such a linking rule will not by itself suffice. In order to apply it, the system needs to know the result of performing the action α in the current state. What is required is a dictionary actions described by the advice-seekers. Presumably some action descriptions, such as undo, repeat, and add-to-a-list, are general and domain-independent, while others, such as send, carbon-copy, respond-to-a-message, and forward are domain-specific.

The performance of any action results in two types of propositions holding: (1) those that have been made true by the performance of the action; (2) frame propositions that were true before the action was performed. (*Frame propositions* are propositions whose truth-value is not affected by the performance of the action.) In most analyses of planning and actions, all frame propositions have equal status, and so either all frame propositions would be part of the resulting state of affairs (in which case the state of affairs would be a complete possible world), or else none would. However, when someone wants to perform an action, he will be more concerned with some frame propositions than with others.

Consider, for instance, an MM user who has finished his message and now wants to move from create mode to send mode. Not only does he want the mode to change, but it is essential to him that his message remain unchanged; after all, the reason he is shifting into send mode is to send the message he's just created. On the other hand, it probably does not matter to him whether his terminal display remains similarly unaffected. A plan that enables him to get to send mode but changes his current message will be unsatisfactory to him, whereas one that gets him to send mode but clears the display will suffice.

When the expert applies Linking Rule 1 to determine the advice-seeker's potential goal, that goal should include those properties that are essential to have hold true. A major aim of the ongoing research is the development of a representation that distinguishes between essential and inessential effects of actions. A preliminary attempt at modeling this distinction is given in [10].

The other query types studied also have associated linking rules. The following dialogue fragment exemplifies a query that is represented by the target predicate wantState(A,p).

A: "How do I get out of read mode and into send mode?"

E: "If you've finished reading all the messages you said to read, a carriage return will get you out. If you want to stop early, just type QUIT."

In this case the associated linking rule is extremely simple:

Linking Rule 2: If an advice-seeker A describes some state of affairs p that he wants (wantState(A,p)), then p is a potential goal of his.

This rule equates the expressed goal with a potential goal. As we shall see, it is only potential, and may not be the domain goal that an appropriate response addresses.

An example of a query represented by wantModifiedAction(A,α,q) is the following:

A: "How can I use the MOVE command to put a mail message in another file but not have it deleted from the MM file?"

E: "Use COPY instead of MOVE."

The action α here is "move a message m"; the modification q is "have m remain in the MM file." The relationship between the effect of the described action and the requested modification will affect the answer given. In the "normal" case, exemplified by the dialogue above, action α does not achieve q, but it does achieve at least some propositions that are compatible with q. Let us call those compatible propositions p'. So for this example p' is "have the text of message m in a file in A's directory." A potential goal of the advice-seeker is p' $\wedge$ q.

Thus a formulation of the linking rule for this query type is

Linking Rule 3: If A says he wants an action α and a modification q to that action, and α does not achieve q but it does achieve at least some propositions p' compatible with q, then p' $\wedge$ q is a potential goal of A.

It turns out that this rule will not properly handle situations in which the advice-seeker is mistaken about the relationship between the result of the action α and the modification q. It not work, for instance, if α itself achieves q. A more general statement of it is given in the following table, along with a summary of some of the other linking rules formulated to date:

Query Type	Linking Rule
wantAction(A,α)	LR1.If A says he wants action α and the result of performing α in the current state of affairs is p, then p is a potential goal of A.
wantState(A,p)	LR2.If A says he wants a state of affairs p, then p is a potential goal of A.
wantModifiedAction(A,α,q)	LR3.If A says he wants an action α and a modification q to that action, then (i) if α does not achieve q but it does achieve some propositions p' compatible with q, then p' $\wedge$ q is a potential goal of A; (ii) if α itself achieves q, then A should be told this; (iii) if α achieves only propositions incompatible with q, then A should be told this.
wantAction(A,OR(α,β))	LR4.If A says he wants α or β and the result of performing α in the current state of affairs is p and the result of performing β in the current state of affairs is q, then the intersection of p and q is a potential goal of A.
wantState(A,AND(p,q))	LR2.(see above)

Inferring Alternative Goals

Consider again the advice-seeker in the introduction who wanted to know how to undo a Control-Z. This query would be represented in the framework as wantAction(undo(Control-Z)). Let us consider what a system based upon the framework would do with this input. First, it would attempt to apply Linking Rule 1, which is associated with the target predicate WantAction. To do so, it would have to look up the definition of *undo*, which would be encoded roughly as follows:

If action α achieves state p when performed in state q, then *undo*(α), when performed in state q, achieves state p.

To instantiate this definition, the expert would next have to look up the definition of the action described by Control-Z, and would find that performing Control-Z in create mode with a certain message m results in send mode with that same message. Hence the action of undoing a Control-Z from send mode with message m should result in create mode with message m. Linking Rule 1 then implies that this state of affairs—being in create mode with message m—is a potential goal of the advice-seeker.

Unfortunately, if the system at this point attempted to derive a plan to achieve this goal, it would fail, and would have to respond with something akin to "You can't get there from here." There is simply no way in MM to return from send mode to create mode without destroying the current message. The human who actually responded to this query, however, was able to provide a much more appropriate answer. She told the advice-seeker "Type EDIT to enter the editor. Then you can finish building your message and, when you're done, type Control-X Control-Z to return to send mode." To arrive at this answer, the human expert had to deduce that the reason the advice-seeker wanted to be in create mode was to finish building his message prior to sending it. He believed that being in create mode was prerequisite to being in send mode with a completed message.

An expert often needs to make just this sort of deduction. It is not always sufficient for her to determine the advice-seeker's potential goal directly from his query: often she will have to determine why those goals are potential goals, and what other goals they are intended to support. The introduction of a set of rules called *alternative-goal rules* will enable this type of reasoning in the goal-inference framework. Alternative goal rules add to the set of potential goals: they are all of the form "if p is a potential goal, then q is also a potential goal."

Allen and Perrault [1] present a set of rules that interconnect an advice-seeker's possible wants. For example, they propose a rule stating that, if an advice-seeker wants some proposition p and if p is a precondition of some action α, then the advice-seeker may want to perform α (Precondition-Action Rule). Another rule states that, if an advice-seeker wants to perform some action β, then he may want the effect of β (Action-Effect Rule). The alternative-goal rule needed in the Control-Z example is a combination of these two rules. It asserts that if p is a potential goal of the advice-seeker and p is a precondition for some action α, then another potential goal of the advice-seeker is the result of α. I will call this Alternative-Goal Rule 1.

For the Control-Z example, the expert knows that one thing the user can do in create mode is add to his current message. Applying Alternative-Goal Rule 1 leads to the introduction of a new potential goal: being in create mode with some message that consists of the current message with additional text appended to it. Let us call this new potential goal g'. Alternative-Goal Rule 1 can then apply iteratively to its own output: one state achievable from g' is being in send mode with the new (appended) message. This is the state of affairs that the human expert chose to treat as the advice-seeker's domain goal: she provided as a response a plan to achieve it.

The reader has a right to be concerned at this point about the combinatorial explosion that may ensue as a result of applying Alternative-Goal Rule 1. As it now stands, this rule can introduce into the set of potential goals all the effects of any action that has as a precondition any proposition already inferred to be a potential goal. Human experts are not nearly so profligate in hypothesizing potential goals of their advice-seekers. Instead they make extensive use of their knowledge of the domain and of the goals people are likely to hold in it. Similarly, the use of rules such as Alternative-Goal Rule 1 by an automated expert will have to be governed by a control strategy that is expectation-driven. We now turn our attention to such a strategy.

Selecting the Most Appropriate Goal

As was just mentioned, human experts use their extensive knowledge of the domain—their expertise—to guide them in inferring what it is that an advice-seeker may be trying to achieve. They know what sorts of things people are apt to want to do.

In the goal-inference framework, this expertise is represented by a set of plausible goals associated with each state of affairs. Each set is partially ordered according to the likelihood of a particular goal: for example, while being in send mode with a complete message, and being out of MM entirely are both plausible goals for someone in create mode, the former is a more likely goal and consequently would be ranked higher.

The idea of having expectations guide goal inference is not a new one. Allen and Perrault [1] discuss it extensively, although they have only global plans serving as expectations. For them, what counts as an expected plan is independent of the questioner's current state. Furthermore, they do not consider an ordering of possible expected plans.

The following algorithm uses the plausible-goal sets to control goal inference:

- •1. Let G be the set of potential goals, and g a distinguished member of G. Set g equal to the potential goal inferred via the linking rules from the query, and make it the only member of G so far. Assume PLAUS is the ordered set of plausible goals associated with the current state. If g is a member of PLAUS
- •2.Then the system attempts to find a plan to achieve g. If it succeeds,
    - •3.Then that plan is given as an answer.
    - •4.Else (if it cannot find a plan to achieve g) the alternative-goal rules are applied backwards to each member of PLAUS in order of plausibilty to attempt to find one that is an alternative to one of the existing potential goals in G. If one is found,
        - •5. Then the newly inferred state is made a member of G. It becomes g, and the algorithm loops to step (2).
        - •6. Else (if PLAUS has been exhausted) the system reports failure.
- •7. Else (if g is not a member of PLAUS) a dialogue must ensue with the advice-seeker to attempt to determine what he is trying to achieve.

Let us consider how the control algorithm would be applied in the Control-Z example. First g would be set equal to the potential goal derived via Linking Rule 1; g can be described as "create mode with the message equal to m, where m is the current message." To start, g is made the only member of the set G of potential goals. PLAUS is the ordered set of plausible goals associated with the current state, "send mode with message m." PLAUS is examined and g is found to be a member of it (step (1)), so the system attempts to achieve g (step (2)). If fails, and so step (4) is executed and PLAUS examined. The highest ranking member of PLAUS that can be derived via an alternative-goal rule from a member of G turns out to be "create mode with the current message and additional text appended to it." This goal becomes the new distinguished member of G, and the algorithm loops to step (2). Again the system attempts to find a plan that achieves (the new) g, and again it fails, so step (4) is once again executed. This time the goal found via the alternative-goal rules is "send mode with the current message and additional text appended to it." This becomes the new g. Now step (2) is again executed, and this time the system is successful in finding a plan that achieves g. That plan is given as the answer, according to step (3).

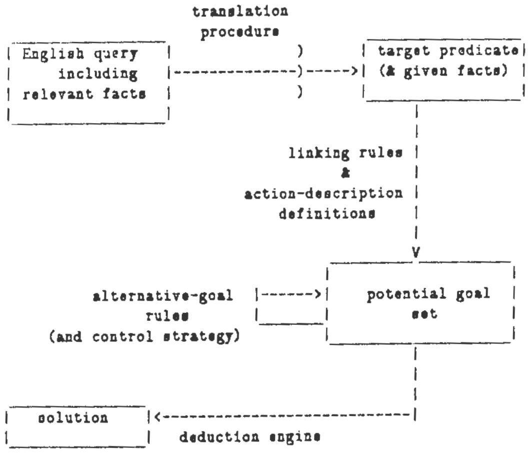
Notice that even if the system finds that the originally inferred potential goal is both plausible and achievable, its behavior may still go beyond that of existing systems in addressing an unexpressed goal. This fact is a result of the reasoning encoded in the linking rules, and it is important to observe because a large number of the collected examples show the expert behaving in just this manner. Recall, for instance, the EMACS example given earlier, in which the advice-seeker asked how to repeat the Control-K command and the expert responded by telling him how to delete a region. No alternative-goal rules need to apply in this example. The potential goal inferred via Linking Rule 1 is found to be plausible (step (1)); the system finds a plan to achieve it (step(2)); and that plan is provided as an answer (step (3)). The linking rule led to the inference of a goal state that was not explicitly expressed in the example, and this goal was planned for independently of the action that was mentioned by the advice-seeker. Even when no alternative-goal rules need be applied, an unexpressed goal may be addressed.

The control algorithm certainly needs some enhancement. One thing, for instance, that must be added is a check that the newly inferred potential goal state in step (5) is not the current state. For the Control-Z example, this would preclude planning to achieve being in send mode with the current message unchanged. Such enhancements to the control algorithm are part of the research now in progress.

Overview of the Goal-Inference Framework

The goal-inference framework has now been described. It is diagrammed below. In the framework, the advice-seeker first presents his query and any relevant facts to the expert. This query is mapped into a target predicate by a set of translation procedures external to the framework. The system then needs to infer the advice-seeker's domain goal before it can provide an answer. It does this in two stages. First, it relates the query to a potential goal. This is done using two types of rules: linking rules and action-description definitions. Linking rules are general rules that relate the type of a query to the type of a potential goal.[c] Action-description definitions are specific rules that capture people's commonly held beliefs about actions.

Once the system has linked the query to a potential goal, it may then have to determine how it might be related to other potential goals of the advice-seeker. Alternative-goal rules are used for this purpose. The application of alternative-goal rules is constrained by an expectation-driven control strategy that makes use of ordered sets of plausible goals associated with the domain states.


translation procedure
English query including relevant facts
target predicate (& given facts)
linking rules & action-description definitions
V
alternative-goal rules (and control strategy)
potential goal set
solution
deduction engine

Conclusion

This paper has described ongoing research to incorporate goal-inference capabilities in expert systems. Study of transcripts of human experts and advice-seekers has shown that providing expert advice is not a one-stage process in which the expert simply attempts to find a solution to the advice-seeker's stated problem. Rather she must first deduce what the actual problem might be, and only then look for a solution. The components of a goal-inference framework have been described, along with a strategy for controlling the inferences. A formalization of the goal-inference process has been attempted [10] using a modal variant of the dynamic logic described by [12], but it proved too cumbersome. At present, several different formal systems for representation are being explored.

Acknowlegements

This work is based on (and largely extracted from) the author's thesis proposal [11]. It was begun while the author was at SRI International, and has also been partially supported by an IBM Graduate Fellowship.

Thanks are due to Barbara Grosz, Aravind Joshi, and Bonnie Webber for all the guidance they have provided. Julia Hirschberg and Kathy McCoy also deserve thanks for hours of discussion about this work.

References

[1] Allen, James F. and C. Raymond Perrault. "Analyzing Intention in Utterances." *Artificial Intelligence 15* (1980), 143-178.

[2] Bruce, Bertram. Belief Systems and Language Understanding. Tech. Rept. 21, Bolt Beranek & Newman, 1975.

[3] Carberry, Sandra. Tracking User Goals in an Information-Seeking Environment. Proceedings of the Third National Conference on Artificial Intelligence, 1983, pp. 59-63.

[4] Cohen, P. R. and C. R. Perrault. "Elements of a Plan-Based Theory of Speech Acts." *Cognitive Science 3* (1979), 177-212.

[5] Finin, Tim and Jeff Schraeger. An Expert System that Volunteers Advice. Proceedings of the Second National Conference on Artificial Intelligence, 1982, pp. 339-340.

[6] Genesereth, M.R. The Role of Plans in Automated Consultation. Proceedings of the 6th International Joint Conference on Artificial Intelligence, 1979, pp. 311-319.

[7] Litman, Diane. Discourse and Problem Solving. 1983. University of Rochester doctoral dissertation proposal.

[8] Mark, William. Representation and Inference in the Consul System. Proceedings of the 7th International Joint Conference on Artificial Intelligence, 1981. pp. 375-381.

[9] Perrault, C. Raymond and James F. Allen. "A Plan-Based Analysis of Indirect Speech Acts." *American Journal of Computational Linguistics 6* (1980), 167-182.

[10] Pollack, Martha E. Generating Expert Answers Through Goal Inference. Tech. Rept. 316, SRI International, 1983.

[11] Pollack, Martha E. Goal Inference in Expert Systesm. Tech. Rept. MS-CIS-84-07, University of Pennsylvania, 1984. Doctoral dissertaion proposal.

[12] Rosenschein, Stanley J. Plan Synthesis: A Logical Perspective. Proceedings of the 7th International Joint Conference on Artificial Intelligence, 1981, pp. 331-337.

[13] Schmidt, C.F., N.S. Sridharan and J.L. Goodson. "The plan recognition problem: an intersection of artificial intelligence and psychology." *Artificial Intelligence 10* (1978), 45-83.

[c] The "type of a query" should not be taken to refer to a syntactic form, but rather to the answers to such questions as: does it express an action or a state?; a conjunction or disjunction?; etc.

USING SPREADING ACTIVATION TO IDENTIFY RELEVANT HELP

Adele Howe*
Timothy W. Finin
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

Abstract: *On-line assistance programs should have the ability to fulfill complex requests for information. We have built an assistance program for the Franz Lisp programming language in which users can enter multiple keyword queries in an unstructured form. The keywords are mapped into the semantic network database, and spreading activation is used to determine the object to be retrieved.*

A many-to-many mapping between keywords and topics permits familiar words to refer to potentially unfamiliar and diverse topics; for example in Franz Lisp, the keyword 'add' is associated with CONS, APPEND and PLUS. A weighting scheme assigns a value for relatedness between keywords and objects, making ADD most closely related to PLUS. Activation is directed by assigning weights to the topics and to the classes of links between objects.

Introduction

On-line assistance programs are frequently included in interactive systems to make them easier to use. Assistance is generally initiated by an explicit request from the user, who enters a command like *help* or *man* [8]. One of the most common and frustrating problems with most conventional help systems is a variation on the old dictionary-lookup problem:

> *How can I look a word up in the dictionary to discover its spelling if I don't know how to spell it?*

Users of computer systems are often faced with a need to learn about some aspect of the system but do not know what information to ask for or exactly how to ask for it.

There are several possible approaches to this problem. One approach is to index information in the Help database by function terms that the user will have a good chance of knowing [8]. Another is to endow the help system with a model of its users which can be used to predict what information the user will need [4]. Still another approach is to develop a help system which the user can easily explore in a top-down manner to find the information he needs [1]. The information retrieval field [7] has a wide set of stategies for identifying possibly relevant items from a large data base.

In this research, we have followed the general approach of providing the user with a network of chunks of help text connected by a variety of syntactic and semantic links. The user can explore the network to seek the answer to a particular question or more generally browse through the network to discover new facts. One of the primary problems with such a network based help system is providing the user with a mechanism to find an appropriate place in the network from which to begin his exploration. Asking the user to employ a top-down search strategy from a root help node places a large burden on him when the network is large. We have provided a *keyword* access system which the user can use to identify relevant starting places in the network.

In some keyword access systems, the user describes the desired information by specifying a set of unique terms for commands or objects in the system. Accessing information using unique keywords, however, presupposes that the user knows the keywords and their corresponding topics in the system. To request information about the function that adds an atom to the front of a list in Lisp, the user would need to know that it is called *cons*.

Alternatively, using many keywords to refer to objects in the system causes confusion about what the user wants to know. For example, one can *add* numbers to numbers (e.g. Plus) or *add* atoms to lists (e.g., Cons). If the user asks for help about *add*, how does the help program determine what type of adding is of interest? We propose a technique for determining responses in help programs given a network database of help information and at least two keywords as input from the user.

Description of HOW?

Our help facility, HOW?, provides textual information as descriptions, examples, and errors about a subset of the language and programming environment of Franz Lisp. The information is organized in a network, where the nodes are textual descriptions and the named links (*subtopic*, *related topic*, *supertopic*, *example-of*, *errors-from*, *details-about*) are the relations between the concepts that the descriptions represent. The system is entered from Lisp via the function HELP along with any number of keywords. Further information is accessed by choosing from a menu of associated topics, executing designated help commands, or entering a list of one or more keywords in an unstructured form.

Translation of User Request to Program Response

Using a string of words to describe a topic seems to be the most natural method for a human. Given at least two words that refer to concepts in the database and the network configuration, a node can be selected for presentation by using spreading activation [2].

Spreading activation theory models human memory retrieval as activation energy spreading from input nodes across links in a semantic network to an intersection [2].

*** Address is now ITT Advanced Technology Center, Shelton, CT 06484**

Applied in the help network database, this theory provides the basis for retrieving information from the database in response to complex, multiple word queries.

Abstractly, our version of this technique involves spread-to-limit from a starting point of a list of keywords, which refer to nodes. In spread-to-limit, the activation is divided among the initial nodes, multiplied by an attenuator or *spread-decay* value and spread to adjacent nodes (and spread to their adjacent nodes and so on) until the activation is below the *spread-limit*, a threshold value that defines negligible activation levels.

Description of the Algorithm

The algorithm relies on the weighting of the keyword types and the meaning of the links for distributing the activation. Keywords can be of three types, based on how closely they describe their topic. Synonomous or unique keys are given a weight of three, keys that describe a topic very well a weight of two, and secondary or loosely descriptive keys a weight of one. Each type of link (supertopic, subtopic, related-topic, details-of, errors-of, examples-of) also has a weight associated with it. These weights are determined by a combination of the designer's intuition of their relative importance and empirical testing.

The parallel search is simulated by maintaining a queue of active nodes (nodes with activation to spread further) and maintaining two activation levels per node. The two levels are *temp-level* and *activation-level*. *Temp-level* is the level of activation that the topic has received since the last time it spread activation to other nodes. *Activation-level* is the total amount of activation that has been accumulated by it during the current retrieval.

The process is started by extracting the keywords from the user request and setting the initial activation levels of the appropriate topics. The starting network activation of 1.0 units is divided evenly among the input keywords (words from the request that correspond to nodes in the network). The initial keyword activation level is then distributed among the topics, referred to by the keyword, by summing their weights (three, two, or one) and distributing the keyword's initial weight between the topics by percentage of total weight. A symbolic form of the equation used appears in figure 1. The topics are added to the active queue, and normal cycling is begun.

Given: KEYS$= K_1, K_2, K_3 ... K_M$
which map to nodes $N_1, N_2, N_3 ... N_S$
with values V of {3, 2 or 1}

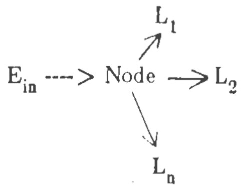
$$\textit{activation/node} = \frac{V_j}{\Sigma V} * W_i$$

where V_j = value associated with each node
$W_i = \textit{activation/key} = \frac{1}{M}$

Figure 1: Formula for Initial Activation Levels

In a normal cycle, the next node on the queue is read. Its *temp-level* is multiplied by an attenuation factor. The attenuation factor is much like the resistance of the links to having energy spread through them; a weak link, such as *errors-from*, has a high resistance and so allows less energy to pass. The attenuation factor reduces the influence of the initial activation over time/distance. If the *temp-level* times the attenuator is less than the *spread-limit*, then no activation will be spread.

The attenuated activation is divided among the associated nodes according to their percentage of the total weight. The weights of the links are summed, and the activation to be spread to each is the input activation multiplied by the weight of the link and divided by the sum of the weights. This amount is added to both the activation-level and the temp-level for the associated nodes, which are pushed onto the queue. Finally, the node just processed is removed from the queue. This formula is in figure 2.



where
A = attenuation factor
W = weight of each link
E_{in} = temp-level for the node

$$E_{out} \text{ for each } L_i = \frac{W_i}{\Sigma W} * E_{in} * A$$

where $(E_{in} * A) >$ *spread-limit*

Figure 2: Formula for Spreading Activation During Cycling

Cycling continues until there no nodes are left on the queue. The *spread-activation* function returns the list of topics, that have been activated, sorted according to their activation-levels. The highest ranking candidate on the list is the topic to be retrieved, with other topics offered to the user as alternatives. A more complex alternative selection scheme would involve choosing alternatives relative to the highest. For example, if the first topic's activation is considerably higher than any other's, then only this topic would be offered. The alternates are printed on the screen for the user's reference and can be accessed by use of a built-in command.

Results and Conclusions

The technique has been tested with a series of multiple keyword requests. Because the system discards irrelevant keywords, pseudo-natural language input is possible, but not necessary. Figure 3 demonstrates a few test cases input to the spreading activation functions. The requests were chosen from the portion of the database that is the most complete.

One benefit of this technique is that a request which seems appropriate to a number of related topics will usually suggest a suitable general node from which the user can explore. This stems from the organization of the network along *supertopic* and *subtopic* links.

```
How do I add an atom to a list?
highest ranking candidate: APPEND
alternatives: APPEND1 CONS List-data-type CAR/CDR

How do I add an atom to the front of a list?
highest ranking candidate: CONS
alternatives: APPEND1 CAR/CDR APPEND List-data-type

How do I add an atom to the back of a list?
highest ranking candidate: APPEND1
alternatives: CONS CAR/CDR APPEND List-data-type

How do I add two lists together?
highest ranking candidate: APPEND
alternatives: APPEND1 CONS List-data-type CAR/CDR
```

Figure 3: Spreading Activation Test Cases

The use of spreading activation with the weighted keyword scheme shows promise as a method for processing complex queries to a help program without developing a natural language interface. Because the database structure and keyword set impact the result of spreading activation searches, the database must currently be hand coded. Current research is investigating ways of automating the database construction process to confront this issue.

References

[1] Acksyn, Robert M., McCracken, Donald L.
The ZOG Project.
Technical Report, Dept. of Computer Science, Carnegie-Mellon University, June, 1982.

[2] Collins, Allan M., Loftus, Elizabeth F.
A Spreading-Activation Theory of Semantic Processing.
Psychological Review 82(6), 1975.

[3] Findler, Nicholas V., Ed.
Associative Networks: Representation and Use of Knowledge by Computers.
Academic Press, N.Y.,N.Y., 1979.

[4] Finin, T.
Providing Help and Advice in Task Oriented Systems.
In *Proc. 8th Int'l. Joint Conf. on Art. Intelligence.* IJCAI, August, 1983.

[5] Foderaro, John K., Sklower, Keith L.
The FRANZ LISP Manual
University of California, 1981.

[6] Howe, Adele.
HOW? A Customizable, Associative Network Based Help Facility.
Technical Report MS-CIS-83-14, Computer and Information Science, Univ. of Pennsylvania, 1983.

[7] Salton, Gerard, Ed.
The SMART Retrieval System: Experiments in Automatic Document Processing.
Prentice Hall, Englewood Cliffs, N.J., 1971.

[8] Sondheimer, Norman K., Relles, Nathan.
Human Factors and User Assistance in Interactive Computing Systems: An Introduction.
IEEE Transactions on Systems, Man and Cybernetics SMC-12(2), March/April, 1982.

Temporal Reasoning with Metric Constraints

Thomas Dean
Department of Computer Science
Yale University
New Haven, Connecticut 06520

Abstract

This paper describes a mechanism for dealing with the representation of events and their effects occurring in and over time. The mechanism, which I refer to as a time map manager (TMM), maintains a set of temporal constraints on a partially ordered kernel of events and their projected effects. · Constraints on the persistence of facts corresponding to the effects of events are maintained so as to avoid contradiction as the partial order is modified. The TMM is a self contained unit which subsumes and extends the functions of procedural nets [7].

1 Introduction

Most interesting planning problems involve dynamic issues:

- the world changes (and one would hope that our view of it is modified accordingly)
- plans evolve and goals shift to meet our varying desires and aspirations
- execution deadlines approach (and sometimes are ignored for one reason or another)
- assumptions made at one point during planning are invalidated by new knowledge or by constraints imposed by other plans

Each of these problems require an ability to reason about time. A pragmatic foundation for reasoning about time should be able to capture the notion that actions can modify the persistence of facts which are true in the world. It should be able to use information about the duration and sequence of events so as to infer possible effects. Finally it should provide a representational structure to support planning and reasoning about the causal structure of the surrounding environment.

A good deal of energy has been spent on developing logics of time and representations for actions and events which attempt to capture our intuitions about cause and effect [5] [6] [1] [2]. The objective of this work is to build a naive but pragmatic theory of causality in order to model the world changing about us and our interaction with it. Using such a model a robot should be capable of developing plans to achieve its goals, executing these plans and recovering from the inevitable problems that arise. To illustrate I'll describe a simple problem from the task domain which has motivated much of my current research. The domain involves a mobile robot, which you can think of as an automated forklift truck, and an environment resembling an industrial machine shop or warehouse.

Suppose that the forklift has two general tasks:

- stack all similar unused items (presumably to conserve floor space) and
- clear all unused items obstructing major thoroughfares in the work space.

There are obvious constraints that one could suggest concerning the order in which plans to achieve these tasks are executed. In particular if, say, a hall was cluttered with vacant desks and the forklift was capable of lifting just one desk at a time then certainly it should clear the desks from the hall before stacking other desks upon them. It would also be reasonable to expect it to integrate stacking with clearing where possible.

A planner has to be able to determine what facts are true at a given point in time (e.g. is a desk clear or a thoroughfare free of obstructions). Whether a fact is true or not at a particular point in time may depend upon the order in which certain tasks are performed and upon other events believed to precede the point in question. This paper presents a possible solution to the problem of efficiently maintaining such knowledge.

2 Time Maps

Events occur in time. The sort of events we will be investigating, events involving actions, can also be said to occur over time. It is convenient to associate with a particular occurrence of an event a temporal interval demarcating its beginning and ending, in which the event is said to occur. In the following we will refer to such an event occurence as an event token. An event type on the other hand refers to a description of an event without reference to a particular time or occurrence fitting that description (e.g. "an attempt made on the president's life" describes an event type while "John Wilkes Booth shot Abraham Lincoln on April 14, 1865" refers to a specific event token). Event tokens can be compared temporally by means of the relative positions of their beginning and ending points in a single dimensional space, that of time. From the information that two event tokens meet, overlap, or that one occurs during the other it is often possible to infer a causal relationship between them or, in the case that a causal relationship is already known, deduce possible events to follow.

In addition to events causing other events, events can be thought of as causing certain facts to "become true". Each fact caused by a given event is associated with an uninterupted temporal interval designating (1) the point in time at which the event ostensibly made the fact true (it might have already been true) and (2) the first point in time following (1) at which the fact is known to be false. The event is said to enable the fact to persist over a particular temporal interval which we refer to as a fact token. The beginning and ending of a fact token is often of interest. The fact token spanning this moment and asserting my continued sanity is hopefully not going to end soon. Other events can shorten the duration of a fact token. I might open a window causing a draft because I am warm and someone else might immediately close that window complaining of the cold.

One can hypothesize about a given event occurring at any point in time but the temporal placement of an event affects both the persistence of the facts caused by that event and the persistence of facts caused by other events. Suppose that we are considering the event *TOKEN1* as occurring at a particular point in time. Facts caused by *TOKEN1* affect the persistence of facts caused by events which precede *TOKEN1* and the facts caused by events which follow *TOKEN1* affect the persistence of facts caused by *TOKEN1*.

A planner must be able to reason about how an event will unfold at a particular point in time. In forming a hypothesis about how an event will occur it is necessary to make certain assumptions concerning the temporal context in which the event is to be placed (i.e. the facts which can be said to be true in adjacent temporal intervals). An event *projection* refers to a set of inferences made assuming that an event will occur in a particular interval. A *projection* of an event token E includes:

1. a set of fact tokens $\{F_1, F_2 \dots F_n\}$ representing the effects of E
2. a set of event tokens $\{E_1, E_2 \dots E_m\}$ suggested by causal inference rules to occur given the immediate temporal context of E
3. constraints upon the time of occurrence of the tokens in the sets mentioned in (1) and (2) relative to E
4. *projections* of the event tokens in $\{E_1, E_2 \dots E_m\}$

I may form a projection (or hypothesis about how the future is to unfold) based upon certain fact tokens and my current estimates concerning their duration or persistence. Later I may feel obliged to retract my hypothesis upon learning that one of the fact tokens on which I was depending has been truncated by another event. After opening the window I might expect the room will soon be cool and the air freshened. When the window is shut after so brief a time I must either reconcile myself with the stale air (and my unaccommodating roommate) or do something to restore the flow of fresh air.

A time map manager (TMM) keeps track of the known relative positions of points corresponding to the beginning and ending of fact tokens and event tokens. The TMM maintains a consistent database of fact tokens by limiting the duration of fact token intervals. By consistent we simply mean that no fact token asserting P overlaps a fact token asserting $\sim$P. The TMM allows other programs, under a wide variety conditions, to keep track of the validity of assumptions made on the basis of facts believed to persist over time. For example the assertion that a particular event has occurred can be withdrawn along with its current projection by simply withdrawing support for the assertion. Any other assertions which depended upon this event or any of its derived effects occurring will be called into question as a side effect. In practice this might mean that an "assumption failure" message would be generated annotating the source of the failure and the parties involved. The planner could then take steps to reestablish the the failed assumptions or replan the threatened steps.

It is also quite easy to displace an event in time while maintaining the temporal relationships between the event and the tokens in its associated projection. This means that a cascade of causally connected tokens can easily be hypothesized to occur at different points in time. Shifting an event from the point in time in which its current projection was formulated to some other point in time may threaten the validity of that projection. For example I may have predicted that the circus clown on the trapeze would be hurt by a fall until I discover that the clown's act will follow the high wire act prior to which a safety net will be set up under the trapeze. The TMM was designed to assist a planner in keeping track of the validity of predictions based upon temporally dependent facts.

3 Time maps in the abstract

The general ideas behind time maps and time map maintenance are quite simple. They rely upon the notion of temporal dependence. A time map designates a partial order on points. Intervals are described as ordered pairs of points with the stipulation that the first point in a pair precedes the second one. Each pair is associated with a token and certain tokens are distinguished by their designating a (temporally dependent) fact. The maintenance algorithm simply sees to it that if there exists a pair *(begin1 end1)* referring to a fact P, then for all pairs *(begin2 end2)* referring to $\sim$P it is not the case that either $\mathit{begin2} < \mathit{begin1} < \mathit{end2}$ or $\mathit{begin1} < \mathit{begin2} < \mathit{end1}$. When such an overlap is detected the token occurring earlier is shortened so that it ends before the later token begins.

If facts were added to the time map and never erased or moved about then enforcing this invariant would be simple. The problem is that we are anticipating the needs of a planning algorithm which will use the time map, and those needs dictate the flexibility to quickly add, shift, remove and restore arbitrarily large event projections. In order to achieve flexibility the maintenance system must keep track for each fact token asserting P, those fact tokens asserting $\sim$P that are "likely" to change position in the temporal partial order.

Determining a reasonable context of events for planning or other forms of problem solving is often a significant part of the problem. Suppose that I'm trying to reconstruct yesterday's events during which my office was entered and my coffee cup with the broken handle and chipped lip stolen. I might be justifiably interested in the event in which I went to the cafeteria, carelessly leaving the office door unlocked behind me. But its not likely that the events corresponding to my waking yesterday morning or riding the shuttle last night will be of any use in discovering just who might have perpetrated the crime.

A practical TMM might designate temporal windows or a set of categories for determining the sort of events that might be considered valuable during planning. Unfortunately fixed-duration windows are likely to prove too restrictive and it is difficult to specify reasonable categories that capture what we mean by relative importance. The duration of an event is certainly not a reliable indicator: the bomb detonations at Hiroshima and Nagasaki together spanned at most a few milliseconds though their repercussions are likely to extend well into the next millenia (if we're so lucky). On the other hand the duration of an event's effects is no indication either, otherwise every person's death would would be considered as an event on a cosmic scale. Rather than introduce events on the basis of some inherent property of the events themselves it seems more reasonable to leave it up to the planner to establish criteria for inclusion.

The solution that I have adopted is to keep track of a map *kernel*, the set of events that the TMM is currently operating on. It is assumed that any events which the planning program deems relevant will be introduced into this kernel. As long as an event remains active in the kernel its projected effects will be taken into account by the consistency maintenance algorithm.

At this point it is obvious that knowledge about events occurring in time can become available in at least two forms in the data base. There are the privileged few events currently residing within the kernel: privileged in the sense that these events, in isolation at least, present to the planner a consistent world model. Then there are all those events that we are not currently concerned with and among which we might find inconsistent temporal assertions. This introduces two additional problems which we will briefly discuss.

The first problem has to do with the status of events lying outside the kernel. How are facts caused by events no longer in the kernel retrieved efficiently from memory? The answer is, that they may not be efficiently retrieved. They will have to be searched for and how efficient that search is will depend upon whatever indices are currently available (perhaps through those events currently residing in the kernel). It is up to the planner to index events relative to other events in such a way that given one event E1 it is reasonably simple to find events whose effects may influence our consideration of E1.

The second problem introduced by a privileged kernel has to do with removing items from the kernel which are no longer relevant to the planner's immediately active tasks. The routine which performs the removal of a token from the kernel has to perform some rather complex juggling in order to maintain all active assumptions in the network. An event token must be disentangled from its projection in such a way as to maintain dependency relationships and yet reduce the size of the kernel. A set of token extraction routines which manage this operation are described in [3]).

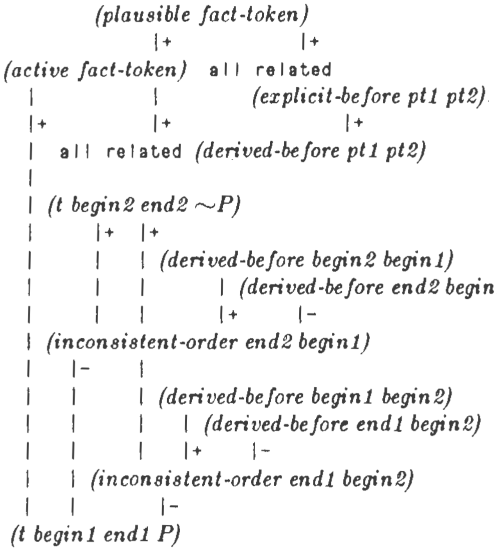
4 Dependency Directed Programming

In order that much of the following discusion be intelligible the reader should be at least passingly familiar with the concept of data dependency [4]. A data dependency network can be thought of as a graph structure whose nodes correspond to beliefs and whose links define support or justificatory relationships. The nodes themselves which we refer to as *ddnodes* (for data dependency nodes) might be associated with just about anything but for our present purposes suppose that each ddnode is associated with a predicate calculus formula (an assertion or implication). In the following we will often refer to the ddnode and its associated formula interchangeably. A ddnode is said to be IN (in our case the associated formula is believed to be true) if there exists a well founded or non-circular justification whose associated ddnodes are themselves IN, or OUT, depending upon the type of support relationships involved (a justification can depend upon some ddnodes being OUT as well as IN). Otherwise a ddnode is said to be OUT. A formula can be made IN by simply asserting it, in which case its justification is the fact that it is a premiss. In other cases a formula will be IN or OUT depending upon the current status of other formulas in the network.

One important fact about data dependency mechanisms (at least those modeled after Doyle's truth maintenance system) is that they support non-monotonic inference. In data dependency terms, this means that changing the status of an existing ddnode can cause a change in the status of other ddnodes. In particular asserting a new premiss can cause something that was formerly IN to become OUT. For example suppose that my belief that stockpiling nuclear weapons is an effective deterrent to their use is contingent upon my belief that no sane country would risk nuclear retaliation. If I later learn that one or more of the major powers believes that a first strike offensive would enable the aggressor to survive a nuclear exchange virtually unscathed then I might wish to reassess more than a few of my beliefs.

5 A description of the actual TMM algorithms

A great deal of the power of TMM stems from the use of dependency hierarchies. Some of the dependencies used in the TMM are shown in figure 1 (+ and - labeled links indicate that the lower assertion depends upon the upper assertion being respectively IN or OUT. Multiple links indicate that an assertion is dependent upon the conjunction of specified support links).



(plausible fact-token)
+
+
(active fact-token)
all related
(explicit-before pt1 pt2)
+
+
+
all related
(derived-before pt1 pt2)
(t begin2 end2 ~P)
+
+
(derived-before begin2 begin1)
(derived-before end2 begin1)
+
-
(inconsistent-order end2 begin1)
-
(derived-before begin1 begin2)
(derived-before end1 begin2)
+
-
(inconsistent-order end1 begin2)
-
(t begin1 end1 P)

Figure 1: TMM data dependencies

The assertion predicates used in the diagram require a bit of explanation. A fact token referring to P is associated with a *tassertion* in the data base of the form *(t begin end P)* where *begin* and *end* correspond to the beginning and ending of the token interval. A token is *plausible* if it is believed to have occurred or to be going to occur. A token is *active* if it is in the kernel of the TMM. A *tassertion* is a kernel data structure: its status is dependent upon its associated fact token being *active.*

All temporal relationships specified in the kernel are represented by the *derived-before* predicate. The efficient operation of the TMM consistency algorithm relies upon the explicit presence in the data base of the transitive closure of *derived-before* on the set of all begin and end points of tokens in the kernel. When the planner introduces a constraint upon two tokens in the kernel (e.g. that one token begins before the other) appropriate *explicit-before* relations are added to the data base dependent upon the two tokens being plausible and whatever other reasons the planner has for adding this constraint. At the same time the transitive closure on the kernel points is updated so that all resulting *derived-before* assertions are dependent upon the *explicit-before* assertion which gave rise to them and the associated tokens being active in the kernel. The *explicit-before* assertions remain in the data base as long as the related tokens are *plausible* but the *derived-before* assertions are part of the kernel and hence they exist only as long as the related tokens are *active.* The system garbage collects ddnodes corresponding to kernel relations (relations such as *derived-before* used exclusively by the kernel maintenance routines) between tokens (or their begin and end points) no longer active in the kernel.

The routine which generates the transitive closure sets up data dependency links on new *derived-before* assertions so that when additions and deletions are made to the data base only the correct subset of *derived-before* assertions will remain. All constraints imposed on the time map partial order are done so in such a way that if at any time in the future support for one of these constraints is withdrawn the data base will "appear" as if the constraint had never been made.

Now we can describe the *tassertion* consistency algorithm. A *tassertion* is added to the data base dependent upon its associated fact token being *plausible.* As long as the fact token remains *active* in the kernel its duration is contingent upon other tokens in the kernel. At the time a *tassertion (t begin1 end1 P)* is added a check is made of all currently active *tassertions* of the form *(t begin2 end2 ~P).* For each such latter *tassertion,* if *(derived-before begin1 begin2)* is IN then it must also be the case that *(derived-before end1 begin2)* is IN and similarly if *(derived-before begin2 begin1)* is IN *(derived-before end2 begin1)* must also be IN.

The *tassertion* initialization process makes the appropriate changes, if necessary, by constraining the kernel points. It then creates a set of *inconsistent-order* assertions which are initially OUT but whose justifications are set up so as to capture the *derived-before* mandates just stated. Remember that the TMM is really only interested in the persistence of fact tokens: that is to say the position of their endpoints in the time map partial order. The *inconsistent-order* assertions refer to constraints which must be imposed if certain conditions become true.

In order to detect and correct inconsistencies, a forward chaining rule is set up in the data base to call a function to restore consistency whenever an assertion of the form *(inconsistent-order ?point1 ?point2)* changes its status from OUT to IN. Thus whenever a consistency mandate is violated its associated *inconsistent-order* assertion will become IN and consistency can be restored as during initialization. With this algorithm a *tassertion* need only be "aware" of *tassertions* added before it, as *tassertions* added afterward will take care of any interactions which they cause.

6 Example

Suppose that the robot is clearing some old desks stacked in *hall14* and it is currently considering two tasks *(unstack desk13 deskstack1)* and *(transport desk17 storeroom3)* where the first is constrained to precede the second. Suppose further that there is a *tassertion (t begin1 end1 (clear-path hall14))* which is currently believed (its associated ddnode is IN) and there is presently no constraint on *end1.* The planner expands *(transport desk17 storeroom3)* resulting in a subtask *(traverse-path hall14 desk17)* beginning at *begin2* and ending *end2.* This expansion includes the addition of the assertion *(passume (traverse-path hall14 desk17))* justified by *(derived-before begin1 begin2)* being IN and *(derived-before end1 end2)* being OUT. That is to say one of the preconditions of using the *traverse-path* plan is that the chosen path be clear throughout the period of traversal (see figure 2 for the setup thus far).

Now suppose that the planner (blindly) expands *(unstack desk13 deskstack1)* resulting in the subtask *(place-at desk13 hall14).* One projected consequence of *(place-at desk13 hall14)* is the *tassertion (t begin3 end3 (not (clear hall14)))* where *begin3* corresponds to the end of the token associated with *(unstack desk13 deskstack1).* The *tassertion* consistency machinery then adds *(explicit-before end1 begin3)* thus truncating the persistence of *(t begin1 end1 (clear-path hall14))* and (among other things) making *(derived-before end1 end2)* IN in the process of updating the transitive closure. The fact that this threatens the *(traverse-path hall14 desk17)* task is noticed by a demon rule primed to watch for assertions of the form *(passume ?task-description)* changing from IN to OUT. The demon body, having its condition satisfied, prompts the planner telling it that a task expansion is in danger of failing and in order to resolve the conflict it can either retract the current plan for *(transport desk17 storeroom3)* or modify the plan that resulted in *(t begin3 end3 (not (clear hall14)))* namely *(place-at desk13 hall14).*

```
begin1      (clear-path  hall14)              end1
 |-------------------------------------|
 (unstack desk13 deskstack1)
   |-----------|
                    (transport desk17 storeroom3)
                      |-------------|
                        (traverse-path hall14 desk17)
                         |--------|
                    begin2        end2
```

Figure 2: Simple task network

7 Related Work

The approach described above subsumes and extends the functionality of Sacerdoti's procedural nets and the attendant mechanism for detecting interactions (using a table of multiple effects (TOME)). The machinery for detecting interactions is more flexible than either NOAH or its descendent NONLIN [8] in that tokens or token complexes can be moved about in the kernel without missing potential interactions due to the changed ordering.

Allen's work [2] comes closest to the issues addressed in this paper. Allen describes an interval based temporal logic and a computational mechanism for handling relative (non-metric) interval relations. The system computes the transitive closure of a set of interval relations and stores the resulting n^2 interval-interval relation assertions in an array for easy access. Reference intervals serve a purpose similar to the kernel in controlling the computation involved in adding and removing relations. Allen also employs reference intervals to organize the temporal data base hierarchically. He mentions but does not discuss in any detail the possiblity of handling the persistence of facts using data dependencies. Comparison between the point based approach taken in this paper and Allen's interval based approach should prove interesting as both systems are developed to deal with significant problems.

8 Application and Conclusions

The TMM algorithms are being used in a hierarchical planning program for solving problems in the forklift domain. The planner uses dependency relations based on the status of *derived-before* assertions to detect and resolve conflicts between plans. Interleaved planning and execution is made possible using routines for removing no longer relevant tokens (and their respective projections) from the time map kernel. Current work is focussed on integrating metric constraints and coordinating the time map representation with a coherent and appropriately indexed representation of episodic memory.

Acknowledgements

Many of the ideas presented in this paper came up in discussions with Drew McDermott, Dave Miller, Yoav Shoham, Stan Letovsky and Eric Gold of the Yale spatial reasoning research group. Special thanks are due to Drew McDermott for his patience in trying to help me articulate my thoughts and urging me in the direction of interesting research problems.

This work was supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored under the Office of Naval Research under contract N00014-83-K-0281.

References

[1] Allen, James.
A general model of action and time.
Technical Report TR97, U. of Rochester Department of Computer Science, 1981.

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

[3] Dean, Thomas.
Time Map Maintenance.
Technical Report 289, Yale University Computer Science Department, 1983.

[4] Doyle, Jon.
A truth maintenance system.
Artificial Intelligence 12(3):231-272, 1979.

[5] McDermott, Drew V.
Planning and acting.
Cognitive Science 2(2):71-109, 1978.

[6] McDermott, Drew V.
A temporal logic for reasoning about processes and plans.
Cognitive Science 6:101-155, 1982.

[7] Sacerdoti, Earl.
A Structure for Plans and Behavior.
American Elsevier Publishing Company, Inc., 1977.

[8] Tate, Austin.
Generating Project Networks.
In *Proc. IJCAI 5.* IJCAI, 1977.

Optical Phenomena in Computer Vision

Steven A. Shafer*
Computer Science Department
University of Rochester
Rochester, New York 14627

Computer vision programs are based on some kind of model of the optical world, in addition to whatever significance they may have in terms of human vision, algorithms, architectures, etc. There is a school of research that addresses this aspect of computer vision directly, by developing mathematical models of the optics and geometry of image formation and applying these models in image understanding algorithms. In this paper, we examine the optical phenomena that have been analyzed in computer vision and suggest several topics for future research.

The three topics that have received the most attention are shading (and glossiness), color, and shadows. Shape-from-shading research, while producing many interesting algorithms and research results, has primarily been based on very simplified models of glossiness. Since realistic gloss models exist within the optics community, we can expect improved computer vision algorithms in the future. Color work in the past has similarly concentrated on developing sophisticated algorithms for exploiting very simple color models, but a more realistic analysis technique has recently been proposed. Shadows have been used by a number of people for simple analysis such as locating buildings in aerial photographs, and a more complex theory already exists that relates surface orientations to shapes of shadows in the image.

A number of problems plague this kind of research, however, including the current inability to model real complexities of illumination and reflection, and the nagging feeling that humans don't seem to rely upon very quantitative analysis of optical properties of materials and illumination. These questions are also addressed.

1. Introduction

Any effort in computer vision can be evaluated on several grounds, such as:

- *Computational* -- What are the algorithms, data structures and architectures involved?
- *Perceptual* -- How well does this work explain or correspond to human visual performance?
- *Semantic* -- What kinds of knowledge are being used, and how do various knowledge sources interact?
- *Analytic* -- What are the underlying geometric and optical models of the world and the imaging process?

Various research efforts have addressed one or more of these sets of issues; for example, the "connectionist" workers study architectures for modeling human vision (computational and perceptual issues [20]). Because the computational aspect of computer vision most closely follows the lines of traditional computer science, it perhaps receives the most attention. But, any or all of the above factors may be crucial in evaluating research ideas and practical performance; thus, the best analytic model may be useless if embedded in a poorly designed algorithm, and at the same time, a sophisticated algorithm based on naive imaging models may never achieve its potential.

In this paper, we examine the analytic aspect of computer vision. This is comprised of a set of geometric and optical models of illumination, reflection, and imaging that provide constraint in performing low-level vision tasks.

There is a definite pattern of evolution in analytic computer vision. Each optical or geometric phenomenon is (or was) historically considered to be first a "source of noise", interfering with perfect and simple images. Eventually, the regular behavior of the phenomenon is studied in analytic computer vision research, and good models are developed. When the models are good enough, the research issue then becomes how to find this phenomenon reliably in real images, and the research becomes computational rather than analytic in nature. Several phenomena, primarily geometric ones, are based on simple enough mathematics that they have already undergone the change to computational problems. They include stereo matching [2], perspective and texture gradients [3, 46, 49, 50], blocks-world line labeling [56], motion [59, 94], and optical flow [18, 37, 73].

In this paper, we are instead concentrating on those phenomena for which good optical models are still under development. Some of these have received considerable attention, such as shading, color, and shadows, which will be the focus of our attention. Several other topics have received limited attention but need more. A few topics have received little or no attention in computer vision to date, and are still considered "noise" even by researchers in analytic computer vision.

The focus of this paper is on models that might be useful for "general vision", i.e. vision in the domains in which humans typically operate. Thus, there will be no substantial discussion of structured lighting techniques or range finders, for example.

2. Shading and Gloss

Early work in image segmentation was generally based on the assumption that pixels representing a single surface should have approximately the same intensity, and that pixels on different surfaces should have different intensities. The first optical

(*) The author's permanent address is: Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213.

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), monitored by the Office of Naval Research under Contract N00014-82-K-0193.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

modeling in computer vision addressed this issue by recognizing that highlights and shading are normal phenomena rather than aberrations in the image.

2.1 Shape From Shading

For fixed directions of illumination and view and a specific surface material, the amount of light reflected from the surface depends on the orientation of the surface. We can denote surface orientation using the gradient space -- *p* represents the degree of left-right slant in the surface, and *q* represents the up or down slant (figure 1) [55, 80]. Horn's *reflectance map R(p,q)* can then be used to represent pixel values as a function of surface gradient (figure 2) [32].

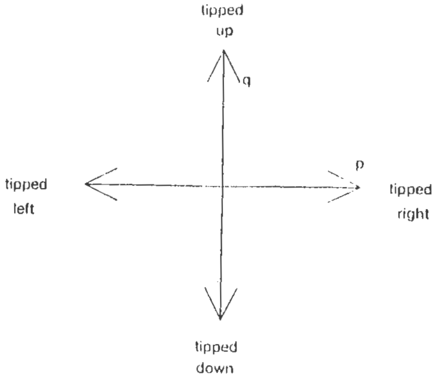


Figure 1: Gradient Space Represents Surface Orientation

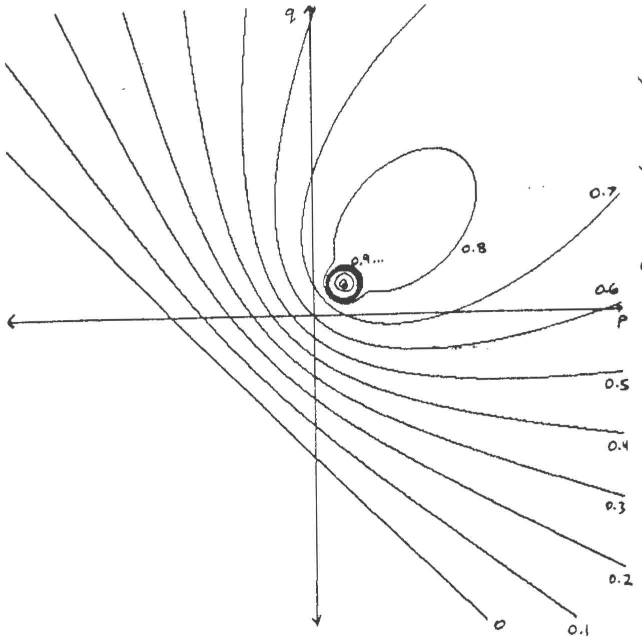


Figure 2: The Reflectance Map Relates Pixel Value to Gradient

The reflectance map provides an explicit relationship between reflected intensity and imaging geometry. A reflectance map can also be expressed in terms of the photometric angles (figure 3) [32]:

- *angle of incidence, i* -- the angle between the illumination direction ***I*** and the surface normal ***N***

- *angle of emittance, e* -- the angle between ***N*** and the viewing direction ***V***

- *phase angle, g* -- the angle between ***I*** and ***V***

In gloss modeling (see below), it is useful also to define the direction of perfect specular reflection ***J*** (the direction of mirror-like reflection, which is ***I*** reflected through ***N***), and the *off-specular angle s* between ***V*** and ***J***. A reflectance map assumes constant *g*; the angles *i* and *e* are then functions of *p* and *q*.

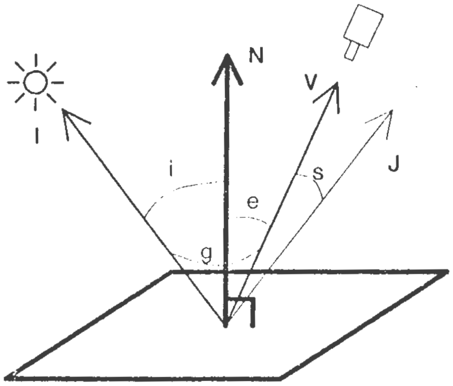


Figure 3: Photometric Angles

Figure 4 shows the reflectance map of a perfect diffuse reflector ("Lambertian surface"), in which *R = cos i*. Such a surface is perfectly matte in appearance -- it exhibits no glossiness (highlights) at all. Most work in shading analysis has been directed towards analyzing Lambertian surfaces (or maria of the Moon, which also have a simple reflectance function [32]).

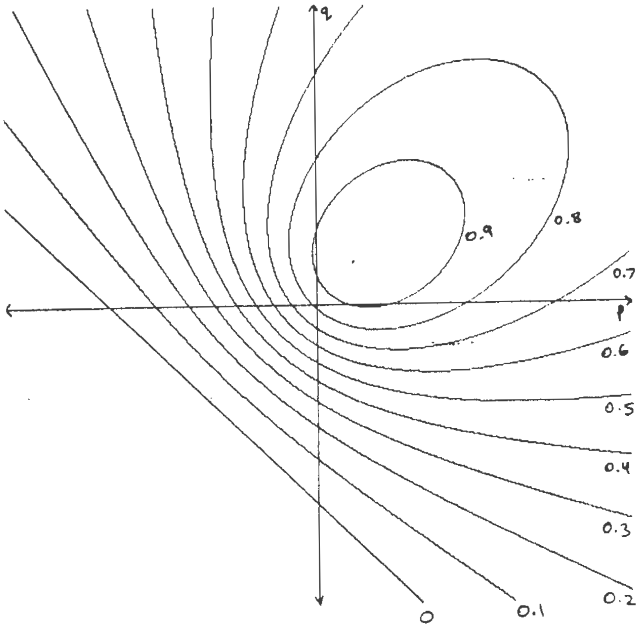


Figure 4: Reflectance Map of a Perfect Diffuse Reflector

When given the intensity of an image at a point, a contour of possible surface orientations in the gradient space is produced, according to the *image irradiance equation I(x,y) = R(p,q)*. This does not give a unique surface orientation at a point, but rather a one-dimensional set of possible orientations. One method for obtaining additional constraint is to use derivatives of *I* and *R*, but the results indicate that some assumptions about surface shape must also be made to obtain unique solutions [10, 12, 31, 68, 69, 89, 104]. Another approach has been to use relaxation with a smoothness constraint on the surface, and possibly some boundary conditions where the surface normal is determined by tangency or shadow edges [4, 11, 31, 41, 101]. Additional constraint can also be provided by taking several images of the same objects with several light sources at different positions using the "photometric stereo" technique [16, 42, 84, 102]. In general, photometric analysis seems to complement such approaches as stereo [33], and photometric arguments have been used to justify surface interpolation between the edges used for stereo disparity measurement [24].

Many of these efforts include a constant term in the intensity relations, intended to model "ambient" light diffusely reflected from the environment. In addition, any such work based on reflectance maps relies on the assumptions built into the reflectance map: orthographic image projection and infinitely distant light source, producing a constant phase angle g.

2.2 Modeling of Glossiness

In most of the work described above, surfaces were assumed to be Lambertian. Real surfaces are not Lambertian, but rather display some amount of glossiness (i.e. highlights). Since very little work has done in the measurement of reflectance maps from real surfaces [30, 35, 103], glossiness is usually taken into account through the use of some *reflection model* that predicts reflection R as a function of the photometric angles i, e, and g (and sometimes s).

When light is reflected from a surface, reflection of two types occurs (figure 5). Some of the light is bounced off of the interface between the air and the surface material, producing glossiness ("specular" reflection); other light penetrates into the material, where it is scattered and may re-emerge ("diffuse" reflection). While diffuse reflection is usually assumed to be Lambertian [96], specular reflection may be scattered about the perfect specular direction J because of the optical roughness of the surface. Various reflection models differ in how they model the distribution of the specular reflection.

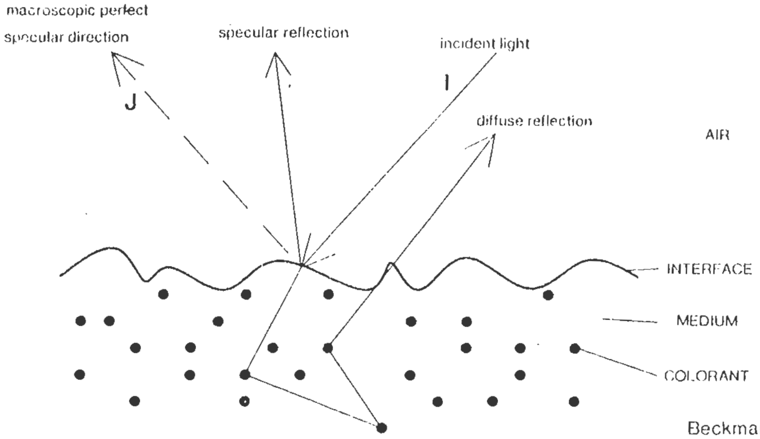


Figure 5: Reflection of Light from a Surface

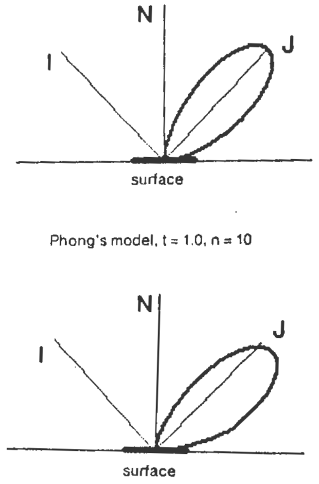
The Lambertian reflection model $R = \cos i$ simply ignores specular reflection altogether. While objects can be coated with special paints that resemble Lambertian reflectors, such techniques cannot be considered suitable for general-purpose vision. Along slightly more general lines, specular reflection can be modeled as occurring only in the perfect specular direction J itself [16, 42]. Such a condition is true only for optically smooth surfaces such as polished optical glass, whereas more typical surfaces are optically rough and exhibit scattered specular reflection.

The most popular model of highlights used in computer vision has been Phong's model intended for computer graphics [71]:

$$R = t\,\frac{n+1}{2}\cos^n s + (1 - t)\cos i$$

In this model, the first term represents the specular reflection and the second term represents diffuse reflection. The material is characterized by t, the total amount of light specularly reflected, and n, the sharpness of the specular peak about the perfect specular direction J. Phong's model has been used for modeling highlights of paint [32] and metal [103], and for finding highlights using intensity gradients [22, 93].

While Phong's model captures some of the aspects of specular reflection, it fails on several counts. It predicts that specular reflection is symmetric about J and that the the spread of the specular reflection for a given material is independent of the angle of incidence i. In fact, specular reflection usually does not have these properties [1]. Phong's model has been widely used because it is relatively simple, although it is not motivated by the underlying physics of reflection [71]. More sophisticated models have been developed within the optics community and adapted for computer graphics use, including Torrance and Sparrow's model of surface facets [92] Beckmann's more general model [5], adapted for computer graphics by Blinn [8] and by Cook and Torrance [17], respectively. These models, unlike Phong's, are based on a consideration of physical reality.



Phong's model, t = 1.0, n = 10

Beckmann's model, t = 1.0, m = 0.2

Curves show amount of specular reflection predicted at various angles for surface illuminated at i = 45 degrees

Figure 6: Gloss Models of Phong and Beckmann

Beckmann's model is based on a statistical description of the probability distribution of surface heights and slopes. When combined with diffraction-theoretic equations for scattering of electromagnetic waves, a reflection distribution function results. The equation used by Cook and Torrance is:

$$R = \frac{t\, f\, k\, \exp\{-\tan^2 n / m^2\}}{\pi\, m^2 \cos^4 n \cos i \cos e} + (1 - t)\cos i$$

with

$$k = \min\left(1, \frac{2\cos n \cos e}{\cos g/2}, \frac{2\cos n \cos i}{\cos g/2}\right)$$

f = Fresnel's reflection coefficient (approx 0.04)
n = angle from ***N*** to bisector of ***I*** and ***V***
("second off-specular angle" [34])
t = parameter: amount of specular reflection (as above)
m = parameter: roughness (typically 0.1 to 1.0)

The models of Beckmann and Phong are compared in figure 6. Beckmann's model has been used in laser speckle studies. It has also been successfully applied in computer vision to detecting defects in metal castings as the basis for segmentation by surface roughness [57, 72]. Beckmann's model describes asymetric distributions of specular reflection and changing distribution of specular reflection with the incidence angle i, but more important, it *describes the inter-relationship of these effects* through the surface roughness parameter m. Even though the equation itself is rather unwieldy, it may make possible the study of these properties of reflection that are missed by Phong's model.

Optical models such as Beckmann's describe only specular reflection; there are no comprehensive models yet for diffuse reflection. The Kubelka-Munk theory assumes that it is isotropic (i.e. Lambertian), while scattering theories do not yet model such important effects as the passage of light through the surface-air interface on its way into and out of the material [27].

One of the problems with applying any reflectance model is the determination of the parameters for a given surface. Grimson solved this problem using a stereo pair of images by finding the specular reflection parameters (for Phong's model) where they could be reliably computed, then applying the resulting parameterized model to the entire surface [25]. This kind of approach seems promising and is probably necessary for the general application of sophisticated reflection models.

Another open question is how much precision is necessary in a reflectance model. There are some results that imprecise reflection models (or maps) yield qualitatively correct but quantitatively inaccurate results [41], and some researchers believe that oversimplified models are sufficient (presumably for the purposes they have in mind) [4, 6, 103]. Intensity measurements in images are also imprecise [32], although sensors are improving.

An interesting illustration of the use of different reflectance models occurs in aerial photointerpretation. Synthetic images are created via terrain models and reflectance maps to determine the registration of real images by comparison. Lambertian reflectance maps have been used for this purpose with some success [34, 53]; Shibata et al. used a Lambertian model at first, then adopted a version of Phong's model with an additional term for backscatter (reflection in the direction of illumination), $t_2 \cos^m g$ (where t_2 and m are parameters of the material) [83]. For their purposes, the backscatter was even more important than normal specular reflection in improving their results.

2.3 Summary of Shading and Gloss Modeling

The reflectance map and accompanying image irradiance equation have been the focal point for a good deal of work, mostly in examination of the ambiguity inherent in analysis of irradiance and in surface reconstruction employing photometric analysis combined with smoothness and sometimes surface shape assumptions. Most of this work has been based on very simple optical models of reflectance, and has been limited to orthography with distant light sources.

Gloss modeling is a critical issue in applying this work to real images. While some believe that the current models such as Phong's are sufficiently precise, there is still some impetus for developing better models. Future work along these lines may rely more on models such as Beckmann's that appeal to the underlying physics of reflection. There has also been some limited work on direct measurement of reflectance maps from material samples.

3. Color

Color pictures obviously contain more information than monochrome (black-and-white) intensity images. However, the most obvious methods for taking advantage of this information yield only incremental improvements in the results of computer vision programs.

3.1 Color Imaging and Color Space

A monochrome image forms pixel values p by integrating light at all wavelengths λ, weighting the amount of light $X(\lambda)$ at each wavelength by the responsivity $s(\lambda)$ of the camera at that wavelength:

$$p = \int X(\lambda)\, s(\lambda)\, d\lambda$$

When a color image is formed with a TV camera, several filters are interposed in front of a monochrome camera one at a time. (Alternate image formation systems, such as beam-splitting or color film scanning, are conceptually similar.) A filter can be characterized by its transmittance $\tau(\lambda)$, which tells what fraction of light at each wavelength passes through the filter. Thus, with a standard set of red, green, and blue filters (such as Wratten filters #25, #58, and #47B [51]) whose transmittances are τ_r, τ_g, and τ_b, the color C of a pixel is:

$$\boldsymbol{C} = \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} \int X(\lambda)\, \tau_r(\lambda)\, s(\lambda)\, d\lambda \\ \int X(\lambda)\, \tau_g(\lambda)\, s(\lambda)\, d\lambda \\ \int X(\lambda)\, \tau_b(\lambda)\, s(\lambda)\, d\lambda \end{bmatrix}$$

All of these integrals are evaluated over the set of wavelengths for which the filter's transmittance and camera's responsivity are nonzero. Because a CCD camera is very sensitive to infrared light [13] and gelatin filters do not block this light [51], an infrared-blocking filter is used for color measurements with a CCD camera.

As shown by the above equation, the color imaging process can be viewed as *spectral projection* from the set of all colored lights $X(\lambda)$ to the *R-G-B color space*, which is the set of all values $[r, g, b]$. The color space is shown in figure 7. It is a cube because the camera's response is bounded by some maximum pixel value for each color component. The main diagonal $r = g = b$ is called the *intensity axis*, and corresponds roughly to the various gray levels from black to white.

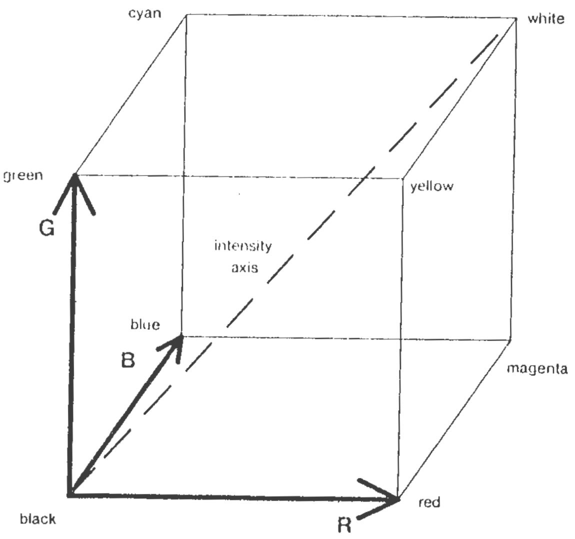


Figure 7: *R-G-B* Color Space

3.2 Color Pixel Classification and Clustering

Most work in color image understanding has been based on the idea that algorithms exploiting pixel differences will work better when more dimensions are avaliable for discriminating among pixel values. One of the heaviest research areas has been color pixel clustering, in which pixels are grouped into sets of related pixels based on distances between clusters of pixel values in color space [9, 15, 60, 65]. The other area that has received much attention is pixel labeling, in which prior knowledge about typical object colors in a particular domain is used to assign object labels to pixels [52, 75, 87, 90, 97, 105]. Such pixel labeling can be very sophisticated and effective, usually depending on how limited the domain is. For example, in aerial photographs, Nagao et al. use typical spectral reflectances to distinguish vegetation, and a common kind of building roof [58]. (N.B. There are so many examples of these same basic strategies that the citations above are representative rather than exhaustive.)

The above efforts are based upon the general idea that, while discrimination among sets of pixels is possible using only intensity information, color provides more dimensions and thus makes clusters of pixels more easily distinguishable from each other. Some attention has been given to finding transformations of the color space that make such clusters even more easily separable [48, 66], but no such set of transformations has been found that decisively improves the quality of image segmentation or labeling based on the above methods.

Another kind of color modeling assumes that the various color components are related to each other and tend to exhibit discontinuities at the same places in the image. This has been exploited by Nevatia, whose color edge finder used the different color components invidually, then looked for places where all three components exhibited edges as an indication of reliability [61]. Similarly, Kanade matched portions of occluded edges based on similarity of color values across the edge pieces [45]. Blicher proposed that two colors are sufficient to perform unambiguous stero matching, but didn't propose an actual algorithm fo doing so [7].

3.3 Analysis of Colored Reflection

There have been a few efforts to analyze color information based on general models of reflection and transmission. For example, shadows have been detected by looking for regions of low intensity adjacent to brighter regions with the same hue [65]. More sophisticated models have included the idea that outdoor shadows tend to be more blue then adjacent illuminated regions because of the blue diffuse skylight [58]. The idea that distant objects tend to be bluish because of scattering of long wavelengths has also been used [87]. Even more sophisticated, but still simple, ideas about color modelling of shadows, etc., were used by Richards Rubin and Richards [76]. They propose that surfaces of differing materials can be recognized by looking for crosspoints in the spectral power distributions (SPDs) $X(\lambda)$ from the two regions of the image.

In recent work, Shafer has proposed a method for breaking down an image into two components: an image of just the glossiness at each point, and an image with all the glossiness removed [82]. This can be done by computing, at each pixel, the amount of specular and diffuse reflection at that pixel. Such analysis is impossible in a monochrome image, where only one value is measured at each pixel, but is theoretically achievable in a color image. It is based on the idea that, while specular reflection is about the same color as the incident illumination (figure 5), diffuse reflection results from interactions with colorant particles and is thus of a completely different color [40, 44]. The resulting images would be useful, for example, in stereo or optical flow situations where the highlights may appear in different places in several images due to camera position changes; the removal of highlights would improve image matching reliability.

Shafer's model expresses two ideas:

1. the total reflected light $L(\lambda, i, e, g)$ is composed of two parts representing the specular reflection L_s and diffuse reflection L_d

2. each of these has a spectral power distribution (SPD) c_s or c_d that gives it a characteristic color, and a geometric scale factor m_s or m_d that tells how this type of reflection varies with the photometric angles:

$$\begin{aligned} L(\lambda,i,e,g) &= L_s(\lambda,i,e,g) + L_d(\lambda,i,e,g) \\ &= m_s(i,e,g)\; c_i(\lambda) + m_d(i,e,g)\; c_d(\lambda) \end{aligned}$$

Using the fact that the color of a mixture of SPDs is the same as the mixture of colors of the individual SPDs (i.e. spectral projection is a linear transform) [79], the above model gives rise to an equation which relates the color $\mathbf{C}$ of any pixel on a surface to the characteristic colors $\mathbf{C}_s$ and $\mathbf{C}_d$ of the specular and diffuse reflection from that surface:

$$\mathbf{C} = m_s \mathbf{C}_s + m_d \mathbf{C}_d$$

Since m_s and m_d vary from pixel to pixel on the surface but $\mathbf{C}_s$ and $\mathbf{C}_d$ do not, this suggests that the distribution of the colors of pixels on a surface will form a parallelogram in color space (figure 8), with $\mathbf{C}_s$ and $\mathbf{C}_d$ as its sides.

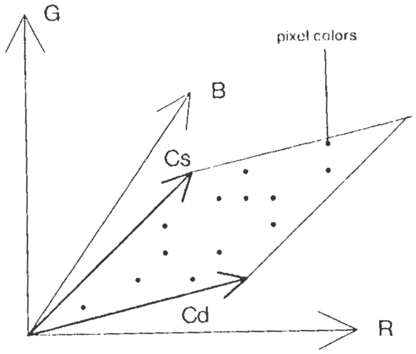


Figure 8: Colors From a Single Surface Form a Parallelogram

The algorithm suggested for exploiting this model is to histogram the colors of a set of pixels in color space, fit a parallelogram in color space to the values, and measure the amounts of reflection m_s and m_d at each point by the position of its color within the parallelogram (figure 9). The model can be extended by adding a term $\mathbf{C}_a$ to represent diffuse (ambient) lighting; in that case, the parallelogram is simply translated by $\mathbf{C}_a$ in color space. Shadow pixels can be recognized as having m_s and m_d both equal to zero, i.e. $\mathbf{C} = \mathbf{C}_a$; this is a much more sophisticated model of shadow colors than simply assuming, for example, that "shadows tend to be bluish".

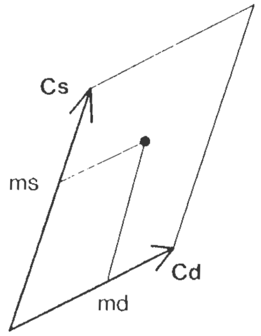


Figure 9: Position In Parallelogram Gives Reflection Magnitudes

While this model is very general, making no assumptions about the size or shape of the light source, the use of orthographic or perspective projection, etc., any such complexities as extended light sources may make the resulting intrinsic images very difficult to analyze. The model also has shortcomings in its slight deviation from the known laws of reflection and the need for prior segmentation, though the former may be negligible and the latter may be addressed by appropriate extension of the model. While this approach has not yet been implemented, it is important because it quantitatively models the relationship between color and scene geometry.

It is interesting to note that the reflection models of the previous chapter are (approximately) instantiations of the color model presented here, with specific functions substituted for $m_s\ (i, e, g)$ and $m_d\ (i, e, g)$.

3.4 Summary of Color Modeling

Most of the work in color image understanding has been exploring clustering and labeling algorithms that exploit very simple color models. Little work has been done in analyzing how color information is related to three-dimensional surface relationships in the scene, although a theoretical approach to this problem has recently been suggested.

4. Shadows

The analysis of shadows primarily involves three processes: finding shadow regions and edges, establishing correspondences between shadow-casting objects and shadows, and geometric analysis of the shadows.

4.1 Finding Shadow Regions and Correspondences

Three different strategies have been identified for identifying shadow regions and edges:

- Finding shadow regions based on intensity and color.
- Finding shadow edges using geometry.
- Identifying shadow edges using intensity correlations.

We will examine each of these topics.

Shadow regions are formed where illumination from the primary light source is blocked by an object. Simple modeling of shadow region intensity might be based on the idea of looking for dark regions in the image. However, since objects themselves might be dark, it is desirable to look for some additional constraint on shadow pixel values. One such constraint is provided by the fact that the diffuse illumination that strikes shadowed regions is related to the color of the light source; thus, shadow regions might be expected to have the same hue as adjacent illuminated portions of the same surface, with lower intensity [65]. When the diffuse light has a different color than the bright light source, this color difference itself can be used. For example, in outdoor photographs, where the sun is yellowish and the sky is blue, shadows tend to look bluish. This observation can be used directly [87] or by looking for "darkness" according to an intensity measure that is weighted towards long (yellow, red, and infrared) wavelengths [58]. Shafer's model of color reflection, presented earlier, makes a more quantitative prediction about shadow colors on individual surfaces.

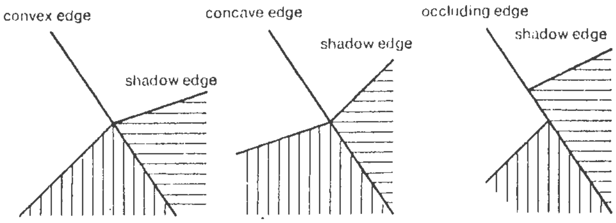


Figure 10: Shadow Edges Bend or Break at Geometric Edges

Another approach to finding shadows is to look for edges that separate light and dark regions in the image. Such edges are likely candidates for labeling as shadow edges. This labeling may be combined with vertex or line labeling schemes [39, 95]. When a shadow falls across two surfaces, the shadow edges bend in one direction or another or break, depending on whether the two surfaces are connected by a convex edge, connected by a concave edge, or not connected (figure 10); these relationships can also be used to identify shadow edges [6].

Finally, shadow edges may be recognized using the variation of image intensity nearby. There are three distinct kinds of shadow edge, each with its own intensity and geometry characteristics (figure 11: shadow-making edges on illuminated polyhedra, terminators of illuminated curved surfaces, and cast shadow edges on shaded surfaces. The first of these, shadow-making edges of polyhedra, can be recognized because they must be convex edges separating an illuminated face from a shaded face of the polyhedron. The second kind of shadow edge, the terminator of a curved surface, is recognizable because the intensity on the shaded side is constant, while the intensity on the illuminated side falls off smoothly from a bright level to the same constant level as the shaded side [4]. Finally, cast shadow edges can be detected because the underlying surface is the same on both sides of the edge; thus, the ratio of intensities on the two sides of the edge should be constant along the edge [54].

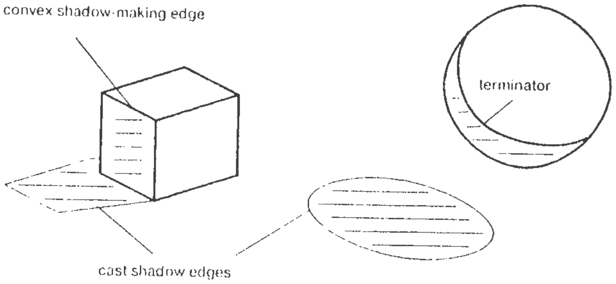


Figure 11: Three Kinds of Shadow Edges

Witkin uses a similar method for distinguishing cast shadow edges from other types of edges [100]. He produces strips of pixels parallel to the edge in question. If the edge is a cast shadow edge, the correlation of these strips is expected to remain constant and high as the edge is crossed, while the "slope" of the intensity function $I(x,y)$ will drop sharply. On the other hand, if the edge is not a cast shadow edge, the correlation will drop while the slope is steady or drops. This is a potentially robust algorithm utilizing the same underlying model of cast shadow edges as described above: that the relative intensity distribution within the shadow region is the same as that of the illuminated portion of the same surface because the surface material is the same.

The correspondence problem between shadows and shadow-casting objects is generally solved using a model of the postion of the light source [38, 54, 64]. Sometimes, the presence of identified objects is used to suggest where shadows might be found [6, 19, 78]; in other cases, the shadows are found first and used to indicate where three-dimensional objects may be found (usually in aerial photographs) [21, 58]. Huertas and Nevatia produced a system for finding buildings in aerial photographs in which shadows and building outlines are used to suggest each other in several ways [39]:

- a sun position model predicts shadows from building positions
- shadow hypotheses arise from two-dimensional vertex types
- intensity histograms are used to confirm shadow hypotheses
- shadows are used to suggest where buildings may be located
- shadows are used to distinguish tall objects from flat ones

4.2 Analysis of Shadows

Once shadows have been located and the shadow-making objects have been identified, shadow analysis can proceed. While most such analysis is geometric, shadows have been used as well for computing the parameters of a model of atmospheric scattering in aerial photographs [85].

Most of the geometric use of shadows has been for identifying the height of objects above a reference plane. In such situations, the size of the shadow (i.e. distance from the shadow-making edge to the cast shadow edge) is proportional to the height of the shadow-making edge above the reference plane. This kind of analysis has been used in manual aerial photointerpretation for many years [88], and has now been applied to computerized aerial image interpretation [6, 38, 39]. In such analysis, the shape of the shadow similarly gives the variation of the height of the object above the reference plane [19]. Related work has used the same method for finding defects in metal castings [67].

The above analysis does not capture all the information available from shadows. Mackworth proposed that a shadow-making edge creates a "shadow plane" containing that edge and the light source; this plane separates the illuminated volume of space from the area shadowed by the object containing the shadow-making edge [55]. This approach was adopted by Shafer and Kanade to produce a theory describing the relationship between shadow edges and surface orientations [81].

Shafer and Kanade began with a "Basic Shadow Problem" involving a single vertex on a shadow-making polygon ***P*** and its associated shadow vertex on a surface ***B*** (figure 12). Information about the orientation (gradient) of ***P*** and ***B*** can be derived from this image using shadow planes. A shadow plane is shown as ***S*** in figure 13; it is a set of light rays, coming from the light source, that graze past ***P*** along the shadow-making edge E_P and strike ***B*** along the cast shadow edge E_B. Edge E_P, joining ***P*** and ***S***, is convex and edge E_B, joining ***S*** and ***B***, is concave. These edge labels give rise to the gradient space relationships shown in figure 13 because two surfaces that are connected by a concave or convex edge have gradients that lie on a line in gradient space perpendicular to the connecting edge in the image [55]. Mathematically, a this provides a one-dimensional constraint on the surface gradients involved. A similar constraint can be found by examining the shadow plane joining the upper edges of ***P*** and ***B*** in figure 12.

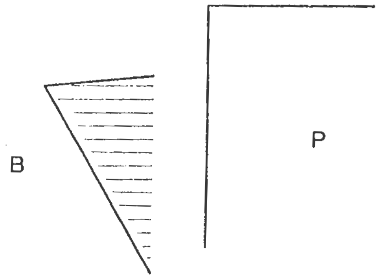


Figure 12: Basic Shadow Problem

The image above provides three constraints, one arising from each pair of shadow-making and shadow edges and one from the vector joining the two vertices, which points at the light source. However, there are six parameters to be computed: the gradient of each of the two surfaces and the direction of illumination (two parameters for each gradient and the illumination vector). Thus, the problem is underconstrained by three degrees of freedom. When the light source is in a different position, the ambiguity is the same; when multiple light sources are present, additional constraint is provided only when the three dimensional direction of illumination is known for each. Since a line drawing with no shadows is also underconstrained by three degrees of freedom [55], shadows do not reduce the ambiguity; instead, they allow information about light source positions to be used to compute surface orientations.

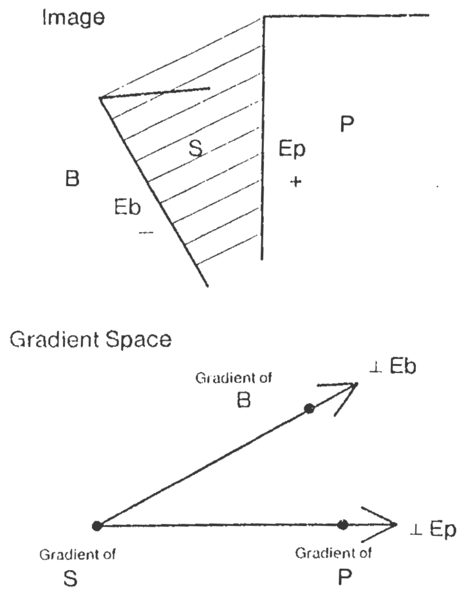


Figure 13: Shadow Plane and Gradient Space Relationship

Figure 14 shows some more complex shadowing situations also discussed by Shafer and Kanade. In figure 14(a), a polyhedron is casting a shadow. In such a picture, the edges marked (*) will be difficult to find because they separate two dark regions. Using shadow geometry, two of these three edges are shown to be redundant and thus unnecessary for the shape recovery of the object. In figure 14(b), a shadow falls on a polyhedron. In this case, as well as the previous case, the additional shadow information balances the missing information concerning the additional surfaces whose orientation is unknown; thus, all such problems are underconstrained by three degrees of freedom regardless of how many surfaces are present. Without shadow analysis, such problems become increasingly complex as more surfaces are added. Finally, in figure 14(c), a curved object casts its shadow on a flat object. Using a derivation similar to that of Witkin [99]. Shafer and Kanade showed that the surface gradient can be determined at every point of the terminator (marked by *) using the shape of the shadow, the position of the light source, and the gradient of the shaded surface (three degrees of freedom total).

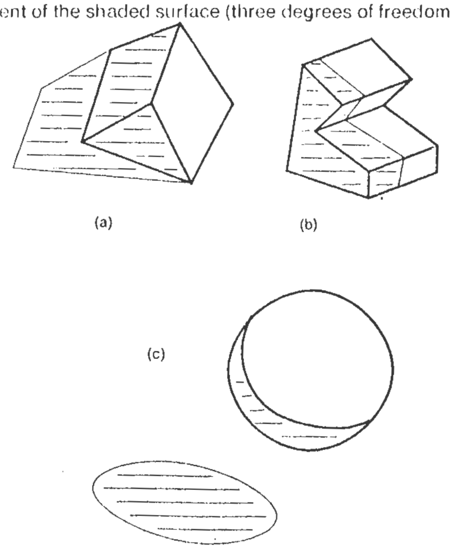


Figure 14: Shadows of Polyhedra and Curved Surfaces

The true significance of the shadow geometry theory lies not only in the mathematical formulas that relate surface gradients to shadow edges, but also in the simple statements that were *deduced* about shadows using this mathematics. Such statements as "multiple light sources add constraint only when their three-dimensional positions are known" are simply not obvious until the mathematics has been developed. Like Beckmann's model of reflectance presented earlier, this kind of theory is useful for increasing our understanding of how light works, quite independent of the value of the formulas themselves.

4.3 Summary of Shadow Modeling

Shadow identification has been primarily based on simple spectral or geometric properties, with some relatively sophisticated methods for shadow edge labelling. The shadow correspondence problem has been approached by using prior knowledge about the position of the sun or other light source. In aerial photographs, tall objects suggest the occurence of shadows and shadows likewise suggest the presence of such tall objects.

Shadow analysis has mostly been limited to determination of the height of an object above a reference plane. A more detailed theory already exists, however, that describes the relationship between surface orientation and shadow shape.

5. Other Optical Phenomena

Gloss, color, and shadows are not the only optical phenomena of interest in computer vision.

5.1 Previous Work

A number of aspects of optical modeling have received some attention in the past in computer vision.

Image sensors induce distortions by nonlinear response to intensity [21, 30, 36], by geometric distortions due to lens design and sensor scanning [30], and by defocussing due to limited depth-of-field [29]. Depth-of-field has actually been used as a source of range information by some researchers [70, 77].

Light sources are really "extended" (with finite area) rather than being points in space. This produces blurred shadow edges and plays havoc with any attempt to determine surface shape using intensity. In an aerial photograph, for example, the edges of shadows cast by airplane wings 7 meters above the ground will be bounded by blurred strips 6 centimeters wide (well below the resolution of typical aerial photographs). Indoors, with windows and light fixtures as light sources, such problems will be far more severe.

Reflection from surfaces is also complicated by polarization and by inter-reflection from multiple surfaces. Polarization of specular reflection can be quite pronounced [43], and this fact has been used to measure surface orientation with a polarizing filter [47]. This work has not been extended to TV camera images, however. Inter-reflection has been studied by Horn [32], who concluded that closed-form analysis appears intractable.

In aerial photointerpretation, models of atmospheric scattering and attenuation of light have been studied [28, 85]. Aerial photointerpretation probably uses the most sophisticated optical models of any branch of computer vision, as we have seen throughout this paper. This is probably due to the relatively limited nature of .the objects being viewed and the existence of very detailed camera, illumination, atmosphere, and reflection models in the remote sensing field [23, 86].

5.2 Looking Ahead

There are a number of optical phenomena that have not been heavily studied but have a direct bearing on the most important of the above theories for gloss, color, and shadow analysis. These are likely areas for future study of optical modeling. They are:

- *Extended Light Sources* -- As noted above, real light sources have finite area and finite distance to the objects being illuminated. Additional study is needed to produce a comprehensive theory of how image intensity is affected by light source shape.

- *Non-Uniform Illumination Distribution* -- Real light sources do not distribute illumination uniformly. Outdoors, the sky is not uniformly bright [14, 85]; indoors, lamp fixture construction contribute to nonuniform light distribution [63]. Intensity analysis must eventually take this into account.

- *Inter-Reflection Among Surfaces* -- As noted above, inter-reflection is very difficult to model. The computer graphics community does have some very coarse models of inter-reflection [17], and some additional thought on this topic is needed in computer vision.

- *Polarization of Specular Reflection* -- Image sensors can be sensitive to polarization in the periphery of the image plane [13]. When this is combined with the polarization of specular reflection, it can be seen that peripheral pixels will represent less contribution of specular reflection than central pixels, for surfaces of certain orientations. The magnitude of this effect is not known, at least within the computer vision community.

- *Extensions to Perspective Projection* -- Most of the above work in optical modeling has been explored only under orthography. In this sense, modeling photometric phenomena lags behind models of geometric phenomena, which have largely been explored in both orthography and perspective. While some attention has been given to reflectance maps under perspective [32, 35], more is needed.

6. The Role of Modeling Optical Phenomena

Optical modeling in computer vision attempts to provide a firm foundation on which to build image understanding algorithms. While this may seem to be a laudable goal, this whole area of research is subject to some controversy and criticism.

There are two related grounds for objection to optical modeling in computer vision. The first may be stated in any of these ways [4, 91]:

- Real images are very complex.

- We do not yet know how to model inter-reflection and extended light sources, but any such modeling appears very difficult.

- Theoretical optical models have only rarely been applied to real images, and those have generally been contrived by special lighting and by painting objects with special paints.

All of the above are true. However, far from being arguments *against* the pursuit of optical models, they may well be interepreted as arguments *promoting* such work. Since optical phenomena tend to evolve from being considered "noise" to being considered "knowledge sources" (as highlights have evolved), the existence of important phenomena that we still consider to be "noise" should be a goad to further research. Rather than concluding that current theories are too complex to be applied to real images, we might conclude that they are far too simple!

The other principal objection to detailed optical models in computer vision might be stated as follows [6, 54, 91]:

- Humans seem to rely on simpler, qualitative models.

- Humans perform vision in complex domains without detailed knowledge of the optical properties of materials and light sources.

- Vision seems to be possible even without quantitative analysis, for example when images are badly distorted.

Here, the objection is to the use of complex formulas in a computer vision program rather than the use of qualitative, intuitive observations about images. Such objections overlook the fact that analyzing detailed models frequently gives rise to insight that can then be described simply. An analogous circumstance in cooking was described by Andy Rooney, an author and television commentator on the "Sixty Minutes" show in the United States [74]. He noted that a good cook looks at a recipe, then puts it away and makes the dish. The good cook doesn't need to consult the recipe line-by-line while he is cooking, because he *understands* the recipe. In computer vision, we derive benefit even if we "put away" the mathematical formulas after deriving simple qualitative observations from them. Such statements might be impossible to make without a deep understanding of the physical or mathematical process involved, and it would certainly be harder to know if (or when) they were true.

Computer vision already seen the evolution towards more sophisticated optical models for metal defect detection and aerial photointerpretation. Our increasing understanding of complex optical phenomena may eventually make such evolution possible for more general vision systems as well.

7. Acknowledgements

Many thanks to Mike Poulin and the staff of the Carlson Library at the University of Rochester for obtaining literature. Comments on this paper were provided by Chris Brown and Takeo Kanade.

8. Bibliography

The interested reader may with to follow the journals *Applied Optics, Journal of the Illuminating Engineering Society, Journal of the Optical Society of America,* and *Proceedings of the SPIE.* Image sensors are described in [13], general radiometry in [26, 62], and appearance measurement (gloss and color) in [27, 40, 44, 96, 98].

1. Barkas, W. W. "Analysis of Light Scattered From a Surface of Low Gloss Into Its Specular and Diffuse Components." *Proc. Phys. Soc. London 51* (1939), 274-295.
2. Barnard, S. T. and Fischler, M. A. "Computational Stereo." *Computing Surveys 14*, 4 (December 1982), 553-572.
3. Barnard, S. T. "Interpreting Perspective Images." *Artificial Intelligence 21* (November 1983), 435-462.
4. Barrow, H. G. and Tenenbaum, J. M. Recovering Intrinsic Scene Characteristics from Images. In *Computer Vision Systems*, Hanson, A. R. and Riseman, E. M., Ed.,Academic Press, New York, 1978, pp. 3-26.
5. Beckmann, P. and Spizzichino, A.. *The Scattering of Electromagnetic Waves from Rough Surfaces.* MacMillan, 1963.
6. Binford, T. O. "Inferring Surfaces from Images." *Artificial Intelligence 17* (1981), 205-244.
7. Blicher, A. P. Stereo Matching from the Topological Viewpoint. Proc. ARPA IUS Workshop, June, 1983, pp. 293-297.
8. Blinn, J. F. "Models of Light Reflection for Computer Synthesized Pictures." *Computer Graphics 11*, 2 (1977), 192-198. SIGGRAPH '77.
9. Broder, A. and Rosenfeld, A. Gradient Magnitude as an Aid in Color Pixel Classification. TR 906, Computer Vision Laboratory, University of Maryland, April, 1980.
10. Brooks, M. J. Two Results Concerning Ambiguity in Shape from Shading. Proc. NCAI, AAAI, August, 1983, pp. 36-39.
11. Brown, C. M., Ballard, D. H., and Kimball, O. A. Constraint Interactions in Shape-from-Shading Algorithms. Proc. ARPA IUS Workshop, September, 1982, pp. 79-89.
12. Bruss, A. R. *The Image Irradiance Equation: Its Solution and Application.* Ph.D. Th., MIT AI Lab, June 1981.
13. Budde, W.. *Optical Radiation Measurements. Volume 4: Physical Detectors of Optical Radiation.* Academic Press, New York, 1983.
14. C.I.E. Daylight: International Recommendations for the Calculation of Natural Daylight. CIE 16 (E-3.2), Commission Internationale de l'Eclairage, 1970.
15. Coleman, G. B. and Andrews, H. C. "Image Segmentation by Clustering." *Proc. IEEE 67*, 5 (May 1979), 773-785.
16. Coleman, E. N. Jr. and Jain, R. "Obtaining 3-Dimensional Shape of Textured and Specular Surfaces Using Four-Source Photometry." *Computer Graphics and Image Processing 18* (1982), 309-328.
17. Cook, R. L. and Torrance, K. E. "A Reflectance Model for Computer Graphics." *Computer Graphics 15*, 3 (August 1981), 307-316. SIGGRAPH '81.
18. Cornelius, N. and Kanade, T. Adapting Optical Flow to Measure Object Motion in Reflectance and X-Ray Image Sequences. Proc. ARPA IUS Workshop, June, 1983, pp. 257-265.
19. Edwards, G. R., Vilnrotter, F. M., Keirsey, D. M., Bullock, B. L., Tseng, D. Y., Close, D. H., Bogdanowicz, J. F., Preyss, E. P., Parks, H. A., and Partridge, D. R. Image Understanding Application Project: Implementation Progress Report. Proc. ARPA IUS Workshop, June, 1983, pp. 156-162.
20. Feldman, J. A. and Ballard, D. H. "Connectionist Models and Their Properties." *Cognitive Science 6* (1982), 205-254.
21. Fischler, M. A., Barnard, S. T., Bolles, R. C., Lowry, M., Quam, L., Smith, G., and Witkin, A. Modeling and Using Physical Constraints in Scene Analysis. Proc. NCAI, AAAI, August, 1982, pp. 30-35.
22. Forbus, K. Light Source Effects. AIM 422, MIT, May, 1977.
23. Goldberg, M., Schlaps, D., Alvo, M., and Karam, G. Monitoring and Change Detection with LANDSAT Imagery. Proc. 6th ICPR, October, 1982, pp. 523-526.
24. Grimson, W. E. L. "Surface Consistency Constraints in Vision." *Computer Graphics and Image Processing 24* (1981), 28-51.
25. Grimson, W. E. L. Binocular Shading and Visual Surface Reconstruction. AIM 697, MIT, 1982.
26. Grum, F. and Becherer, R. J.. *Optical Radiation Measurements, Volume 1: Radiometry.* Academic Press, New York, 1979.
27. Grum, F. and Bartleson, C. J., editors.. *Optical Radiation Measurements, Volume 2: Colorimetry.* Academic Press, New York, 1980. Chapter 7, "Colorant Formulation and Shading", by E. Allen.
28. Haralick, R. M., Wang, S., and Elliott, D. B. Spatial Reasoning to Determine Stream Network from Landsat Imagery. Proc. 6th ICPR, October, 1982, pp. 502-516.
29. Horn, B. K. P. Focusing. AI Lab Memo 160, MIT, 1968.
30. Horn, B. K. P. *Shape From Shading: A Method for Obtaining the Shape of a Smooth Opaque Object From One View.* Ph.D. Th., MIT, November 1970. MAC-TR-79.
31. Horn, B. K. P. Obtaining Shape from Shading Information. In *The Psychology of Computer Vision*, Winston, P. H., Ed.,McGraw-Hill, New York, 1975, pp. 115-155.
32. Horn, B. K. P. "Understanding Image Intensities." *Artificial Intelligence 8* (1977), 201-231.
33. Horn, B. K. P., Woodham, R. J., and Silver, W. M. Determining Shape and Reflectance Using Multiple Images. AIM 490, MIT, August, 1978.
34. Horn, B. K. P. and Bachman, B. L. "Registering Real Images Using Synthetic Images." *Comm. ACM 21*, 11 (November 1978).
35. Horn, B. K. P. and Sjoberg, R. W. "Calculating the Reflectance Map." *Applied Optics 18* (1979), 1770-1779.
36. Horn, B. K. P. and Woodham, R. J. "Destriping LANDSAT MSS images." *Computer Graphics and Image Processing 10* (1979), 69-83.

37. Horn, B. K. P. and Schunck, B. G. "Determining Optical Flow." *Artificial Intelligence 17* (1981), 185-203.

38. Huertas, A. An Edge Based System for Detecting Buildings in Aerial Images. USC Report ISG-101, USC, March, 1982. In Image Understanding Research, R. Nevatia editor.

39. Huertas, A. and Nevatia, R. Detection of Buildings in Aerial Images Using Shape and Shadows. Proc. IJCAI-83, August, 1983, pp. 1099-1103.

40. Hunter, R. S.. *The Measurement of Appearance.* J. Wiley and Sons, New York, 1975.

41. Ikeuchi, K. and Horn, B. K. P. "Numerical Shape from Shading and Occluding Boundaries." *Artificial Intelligence 17* (1981), 141-184.

42. Ikeuchi, K. "Determining Surface Orientations of Specular Surfaces by Using the Photometric Stereo Method." *IEEE PAMI 3*, 6 (November 1981), 661-669.

43. Jenkins, F. A. and White, H. E.. *Fundamentals of Optics.* McGraw-Hill, New York, 1976.

44. Judd, D. B. and Wyszecki, G.. *Color in Business, Science and Industry.* J. Wiley and Sons, New York, 1975.

45. Kanade, T. "Recovery of the Three-Dimensional Shape of an Object from a Single View." *Artificial Intelligence 17* (1981), 409-460.

46. Kanade, T. and Kender, J. R. Mapping Image Properties into Shape Constraints: Skewed Symmetry, Affine-Transformable Patterns, and the Shape-from-Texture Paradigm. In *Human and Machine Vision*, Rosenfeld, A. and Beck, J., Ed.,Academic Press, New York, 1983.

47. Kazutada, K. A Polarimetric Approach to Shape Understanding of Glossy Objects. Proc. IJCAI-79, August, 1979, pp. 493-495.

48. Kender, J. R. Instabilities in Color Transformations. PRIP-77, IEEE Computer Society, Troy NY, June, 1977, pp. 266-274.

49. Kender, J. R. *Shape from Texture.* Ph.D. Th., Carnegie-Mellon University, November 1980.

50. Kender, J. R. Environmental Relations in Image Understanding: The Force of Gravity. Proc. ARPA IUS Workshop, June, 1983, pp. 249-256.

51. Publication B-3. *Kodak Filters for Scientific and Technical Uses.* Eastman Kodak Company, Rochester, NY 14650, 1970.

52. Levine, M. D. and Shaheen, S. I. A Modular Computer Vision System for Picture Segmentation and Interpretation, Part 1. PRIP-79, IEEE Computer Society, Chicago, Ill., August, 1979, pp. 523-533.

53. Little, J. J. Automatic Registration of Landsat MSS Images to Digital Elevation Models. Workshop on Computer Vision, IEEE Computer Society, Rindge, NH, August, 1982, pp. 178-184.

54. Lowe, D. G. and Binford, T. O. The Interpretation of Three-Dimensional Structure from Image Boundaries. Proc. IJCAI-81, August, 1981, pp. 613-618.

55. Mackworth, A. K. "Interpreting Pictures of Polyhedral Scenes." *Artificial Intelligence 4* (1973), 121-137.

56. Mackworth, A. K. How to See a Simple World: An Exegesis of Some Computer Programs for Scene Analysis. In *Machine Intelligence 8*, Elcock, E. W. and Michie, D., Ed.,American Elsevier Pub. Co., New York, 1977, ch. 25, pp. 510-537.

57. Mundy, J. L. and Porter, G. B. Visual Inspection of Metal Surfaces. Proc. 5th ICPR, December, 1980, pp. 232-237.

58. Nagao, M., Matsuyama, T., and Ikeda, Y. "Region Extraction and Shape Analysis in Aerial Photographs." *Computer Graphics and Image Processing 10* (1979), 195-223.

59. Nagel, H.-H. "Representation of Moving Rigid Objects Based on Visual Observations." *Computer 14*, 8 (August 1981), 29-39.

60. Nagin, P. A., Hanson, A. R., and Riseman, E. M. Region Extraction and Description Through Planning. COINS TR 77-8, U. Mass., May, 1977.

61. Nevatia, R. "A Color Edge Detector and Its Use in Scene Segmentation." *IEEE Trans. Systems, Man, and Cybernetics TSMC-7*, 11 (November 1977), 820-826.

62. Nicodemus, F. E., Richmond, J. C., Hsia, J. J., Ginsberg, I. W., and Limperis, T. Geometrical Considerations and Nomenclature for Reflectance. NBS Monograph 160, National Bureau of Standards, October, 1977.

63. Nuckolls, J. L.. *Interior Lighting for Environmental Designers.* J. Wiley and Sons, New York, 1976.

64. O'Gorman, F. Light Lines and Shadows. School of Social Sciences, University of Sussex, Sussex, England, December, 1975.

65. Ohlander, R. *Analysis of Natural Scenes.* Ph.D. Th., Carnegie-Mellon Univ. Computer Science Dept., 1975.

66. Ohta, Y., Kanade, T., and Sakai, T. "Color Information for Region Segmentation." *Computer Graphics and Image Processing 13* (1980), 222-241.

67. Okawa, Y. "Automatic Inspection of the Surface Defects of Cast Metals." *Computer Vision, Graphics, and Image Processing 25* (1984), 89-112.

68. Pentland, A. P. Local Computation of Shape. Proc. NCAI, AAAI, 1982, pp. 22-25.

69. Pentland, A. P. Local Analysis of the Image: Limitations and Uses of Shading. Workshop on Computer Vision, IEEE Computer Society, Rindge, NH, August, 1982, pp. 153-161.

70. Pentland, A. P. Depth of Scene from Depth of Field. Proc. ARPA IUS Workshop, September, 1982, pp. 253-259.

71. Phong, Bui Tuong. "Illumination for Computer Generated Pictures." *Comm. ACM 18* (1975), 311-317.

72. Porter, G. B. and Mundy, J. L. "Automatic Visual Inspection of Metal Surfaces." *Proc. SPIE 281, Techniques and Applications of Image Understanding* (1981), 176-181.

73. Prazdny, K. "On the Information in Optical Flows." *Computer Vision, Graphics, and Image Processing 22* (1983), 239-259.

74. Rooney, A. A.. *And More by Andy Rooney.* Warner Books, New York, 1979.

75. Rubin, S. *The ARGOS Image Understanding System.* Ph.D. Th., Carnegie-Mellon Univ. Computer Science Dept., 1978.

76. Rubin, J. M. and Richards, W. A. Color Vision and Image Intensities: When are Changes Material? AI Memo 631, MIT, May, 1981.

77. Schlag, J. F., Sanderson, A. C., Neuman, C. P., and Wimberly, F. C. Implementation of Automatic Focusing Algorithms for a Computer Vision System with Camera Control. CMU-RI TR-83-14, Carnegie-Mellon University Robotics Institute, 1983.

78. Selfridge, P. G. *Reasoning About Success and Failure in Aerial Image Understanding.* Ph.D. Th., Computer Science Dept., U. Rochester, May 1982.

79. Shafer, S. A. "Describing Light Mixtures Through Linear Algebra." *J. Optical Soc. Am. 72*, 2 (February 1982), 299-300.

80. Shafer, S. A., Kanade, T., and Kender, J. "Gradient Space under Orthography and Perspective." *Computer Vision, Graphics, and Image Processing 24* (1983), 182-199.

81. Shafer, S. A. and Kanade, T. "Using Shadows in Finding Surface Orientations." *Computer Vision, Graphics, and Image Processing 22* (1983), 145-176.

82. Shafer, S. A. Using Color to Separate Reflection Components. Computer Science Department, University of Rochester, in preparation, 1984.

83. Shibata, T., Frei, W., and Sutton, M. Digital Correction of Solar Illumination and Viewing Angle Artifacts in Remotely Sensed Images. Proc. 1981 Symposium on Machine Processing of Remotely Sensed Data, 1981, pp. 169-177.

84. Silver, W. Determining Shape and Reflectance Using Multiple Images. Master Th., MIT EE and CS Dept., 1980.

85. Sjoberg, R. W. and Horn, B. K. P. Atmospheric Modelling for the Generation of Albedo Images. Proc. ARPA IUS Workshop, April, 1980, pp. 58-63.

86. Slater, P. N.. *Remote Sensing: Optics and Optical Systems.* Addison-Wesley, 1980.

87. Sloan, K. *World Model Driven Recognition of Natural Scenes.* Ph.D. Th., U. Penna. Moore School of Electrical Engineering, June 1977.

88. Smith, H. T. U.. *Aerial Photographs and Their Applications.* Appleton-Century, New York, 1943.

89. Smith, G. B. The Relationship Between Image Irradiance and Surface Orientation. Proc. ARPA IUS Workshop, June, 1983, pp. 243-248.

90. Tenenbaum, J. M. and Weyl, S. A Region-Analysis Subsystem for Interactive Scene Analysis. Proc. 4th IJCAI, September, 1975, pp. 682-687.

91. Thompson, W. B. and Yonas, A. What Should be Computed in Low Level Vision Systems. Proc. NCAI, AAAI, August, 1980, pp. 7-10.

92. Torrance, K. E. and Sparrow, E. M. "Theory for Off-Specular Reflection from Roughened Surfaces." *J. Optical Soc. Am. 57* (September 1967), 1105-1114.

93. Ullman, S. "Visual Detection of Light Sources." *Biol. Cybernetics 21* (1976), 205-212.

94. Ullman, S. "Analysis of Visual Motion by Biological and Computer Systems." *Computer 14*, 8 (August 1981), 57-69.

95. Waltz, D. Understanding Line Drawings of Scenes with Shadows. In *The Psychology of Computer Vision*, Winston, P. H., Ed.,McGraw-Hill, New York, 1975, pp. 19-91.

96. Wendlandt, W. W. and Hecht, H. G.. *Reflectance Spectroscopy.* Interscience, New York, 1966.

97. Weymouth, T. E., Griffith, J. S., Handon, A. R., and Riseman, E. M. Rule-Based Strategies for Image Interpretation. Proc. NCAI, AAAI, August, 1983, pp. 429-432.

98. Williamson, S. J. and Cummins, H. Z.. *Light and Color in Nature and Art.* J. Wiley and Sons, New York, 1983.

99. Witkin, A. P. *Shape from Contour.* Ph.D. Th., MIT, November 1980. MIT AI-TR-589.

100. Witkin, A. Recovering Intrinsic Scene Characteristics from Images. Proc. NCAI, AAAI, August, 1982, pp. 36-44.

101. Woodham, R. J. A Cooperative Algorithm for Determining Surface Orientation from a Single View. Proc. IJCAI-77, August, 1977, pp. 635-641.

102. Woodham, R. J. Photometric Stereo: A Reflectance Map Technique for Determine Surface Orientation from Image Intensity. Proc. SPIE 22nd Technical Symposium, vol. 155, August, 1978.

103. Woodham, R. J. *Reflectance Map Techniques for Analyzing Surface Defects in Metal Castings.* Ph.D. Th., MIT AI Lab, June 1978.

104. Woodham, R. J. Relating Properties of Surface Curvature to Image Intensity. Proc. IJCAI-79, August, 1979, pp. 971-799.

105. Young, I. T. "The Classification of White Blood Cells." *IEEE Trans. Biomedical Engineering BME-19*, 4 (July 1972), 291-298.

PROCEDURAL ADEQUACY IN AN IMAGE UNDERSTANDING SYSTEM

Jay Glicksman
Computer Science Laboratory
Texas Instruments
Dallas, Texas

Abstract -- Model-driven vision systems often employ feedback loops as part of their control mechanisms so that the context provided by previous analysis can effect subsequent processing. The MISSEE image understanding system combines a cycle of perception with a semantic network to interpret aerial photographs of urban areas.

The semantic network links nodes via two hierarchies, specialization and composition, and two relations, instance and neighbour. Hierarchical control results in the network being built and the structure of the network in turn influences the cycle of perception. Heterarchical control promotes efficient use of context in a message-passing paradigm.

MISSEE combines information from up to three sources: a digitized image, a sketch map, and advice given by the user. Several scenes have been interpreted and the results are reported.

Introduction

An important aspect of vision systems is the matter of procedural adequacy [1]. Most domain independent systems tend to operate in a bottom-up, linear fashion beginning with the image and working towards the possible objects in it. Domain dependent systems generally have a wider range of control strategies, but most contain either a top-down component or a feedback loop of some sort. Such systems have the problem of deciding which object to hypothesize (the chicken and egg problem [2]) but once determined, these programs can make efficient use of object-specific heuristics to plan their strategies. **A priori** expectations derived from the models as well as the context resulting from previous interpretation both provide important clues (and cues) to guide the understanding of images.

There are two reasons for employing more complex control mechanisms in vision systems. First, by making better use of the available information, the program is able to achieve superior results than would be possible by trying all possible mappings of models to image features. Context-sensitive feature extraction results in more appropriate boundaries being chosen, thresholds selected, etc. The second reason is a question of efficiency. In all but trivial scenes, the combinatorial explosion of possible image to scene mappings requires some selection method to limit the number of choices.

The domain of this research is the area of aerial photo-interpretation of small urban scenes. Objects on the level of roads, rivers, bridges, and towns are postulated from features in the image. These objects are fit into the hierarchies that also contain road-systems, river-systems, waterbodies, and geo-systems. Objects are represented as schemata [3] which contain value slots, relation slots, confidence values, and attached procedures.

Previous Research

Feedback control loops were well exemplified in the HEARSAY II speech understanding system [4]. Processing progressed in a two-stage fashion known as hypothesize and test. In production rule systems, this is done by modifying the global data base.

The VISIONS system [5] uses schemata as the primary form of representation. Schemata guide the instantiation of hypotheses during interpretation. The system follows a three-phase mode of control, focus-expand-verify, which, depending on the knowledge sources involved, could exhibit either bottom-up or top-down control.

In the ACRONYM vision system [6] objects are modeled in three dimensions as generalized cones and their relationships are represented in an "object" graph. The "restriction" graph contains constraints on the spatial relationships between objects. From these representations, a "prediction" graph is generated to hypothesize object to image feature matches. "Observation" and "interpretation" graphs are built during the analysis of an image.

A Cycle Of Perception

Cognitive psychologists have examined the role of schemata as collections of structures and processes that both accept perceptual information and direct movements and exploratory activities [7]. Neisser describes a perceptual cycle where anticipatory schemata determine plans for perceptual action, the outcome of which modifies the original schemata. Its three phases are "schema" directing "exploration" which samples "objects" in the image which modifies the "schema". A similar feedback loop has been proposed specifically for model-based computer vision [8]. It is called the cycle of perception. A slightly modified version can be seen in Figure 1.

Depending on where the cycle is entered, one will observe different modes of operation which correspond to the traditional methods of control in Computer Science. If the cycle begins at the "object" stage then bottom-up processing will result. If the cycle is entered at the stage of schema invocation then top-down behaviour would result. The model knowledge

The research discussed in this paper was carried out at the University of British Columbia.

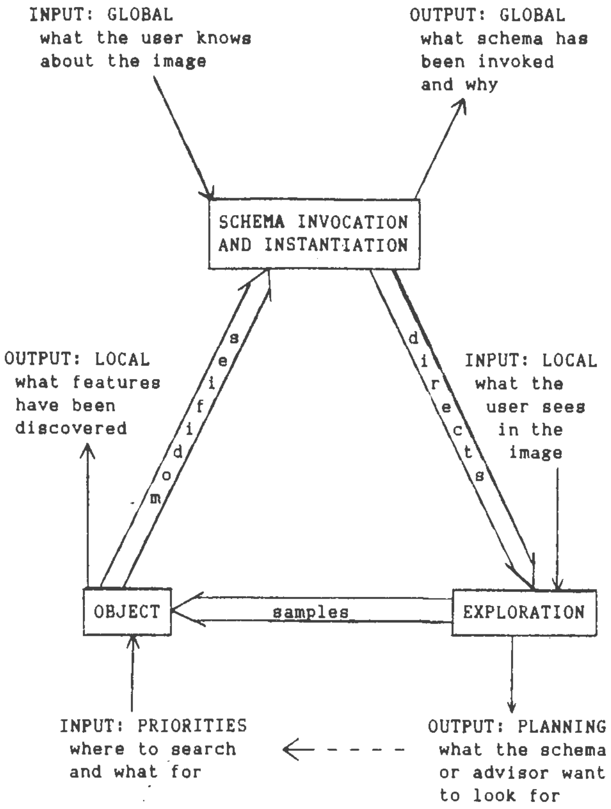


Figure 1. A Cycle of Perception

contained in the schema will be used to hypothesize the existence of objects in the image. There will also be information concerning how and where to obtain the information in the image that will verify the hypotheses. So, the exploration module will cause the image to be sampled to find the appropriate information. Depending on what is found in the image, the schema will be suitably modified. The instantiated schema can then request more information to confirm its existence or it can use the knowledge gained to help hypothesize the whereabouts of other objects.

In this way, the cycle of perception promotes cooperation among the schemata. As objects are instantiated they communicate to other schemata to give advice and/or to build the semantic network. This in turn initiates another loop around the cycle as a new schema is instantiated. The schemata work as individuals (via attached procedures) and together (via messages) to bring about the interpretation of particular information sources.

The cycle of perception is similar to other feedback paradigms such as hypothesize and test. It is particularly well-suited to a schema-based system because it identifies the natural places where schemata can move into and out of the focus of attention. The consequences of this are discussed in the following section.

The Hierarchies

Besides the cycle of perception, control is strongly influenced by the hierarchical relationships between schemata. In an object-oriented system, the connections between the objects determine which pairs of schemata can most profitably communicate. Messages can be sent from any schema to another; however, part of the goal of interpretation is to join all of the schemata into a unified network necessitating significant communication between neighbouring nodes.

The two important hierarchies in terms of control are specialization and decomposition. These are combined with two relations, instance and neighbour, to form the semantic network. Instances represent possibly hypothetical objects particular to a given image. This distinguishes particular from stereotypic knowledge. The neighbour relation is used to group objects in the composition hierarchies. Objects that are neighbours are "part of" the same structure.

Figure 2 shows the generic hierarchies used in the MISSEE system. In graphic form, the standard control paradigms are apparent. Top-down, model-driven behaviour will be observed when schema-specific routines respond to messages from "above" in the hierarchies. Bottom-up, data-driven control comes from below, since the objects at the "bottom" of the graph (road, river) can be considered to be the most primitive, "closest" to the data. The data (the intensity image and the sketch map) can be thought of as being another layer "below" the depicted schemata. The relationship between data and schemata is not specialization or decomposition but some type of mapping. The directions in the graph make sense because the graph has been displayed with the schemata ordered from the bottom by increasing generality and composition.

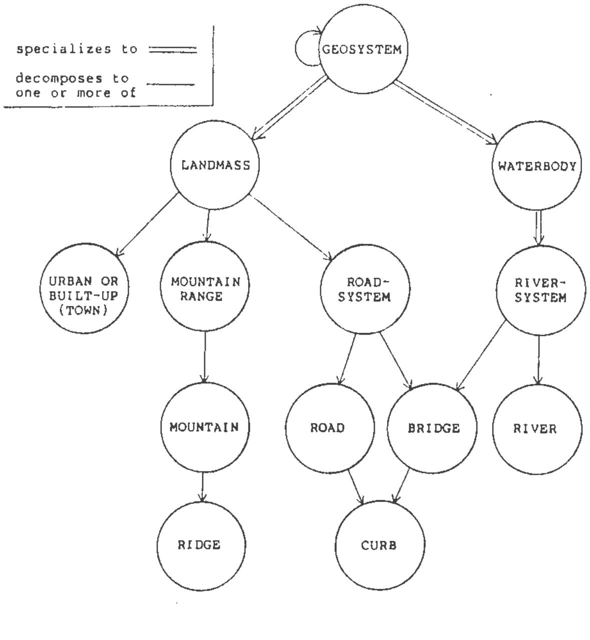


Figure 2. The Generic Objects in MISSEE

Control: Top-Down/Bottom-Up/Middle-Out

Top-down control can take two forms. The first is like top-down parsing: e.g. if one wants to establish a geosystem then find either a landmass or a waterbody. Model knowledge is used at each stage in deciding how to move down the hierarchies until a bottom level schema (terminal symbol) is reached. If it can be instantiated from the information sources then the hierarchies that have been built up can remain in place. Otherwise, control must back up to a choice point and a different branch must be taken.

The other type of top-down control can take effect only after part of the interpretation has taken place. Previous results help form the context of the current stage of processing. The context plus the model knowledge of the schema are used to produce a more efficient search strategy. For example, if it has already been established that a bridge exists, then if control currently resides in the road system schema, a good strategy would be to look for a road going over the bridge. Since the location and orientation of the bridge are known, this greatly constrains the resulting search for evidence of a road.

This second type of top-down, or expectation-driven, control can also be used to relax thresholds. Thresholds are those cutoff points that schema-specific routines must use to determine whether the features in an information source are sufficient to instantiate the object. For example, the intensities of pixels representing water are usually less than those corresponding to land. A threshold for water is an intensity value that classifies as water all pixels having lower intensities. However, it is generally true that if one tries to find an intensity value that divides all pixels into two categories (water and land), there will always be some pixels that are classified incorrectly.

If one starts from a conservative position that will only accept feature values that are reliable indicators of the presence of an object (the principle of least commitment [9]), then as the expectations rise for existence of the object, the thresholds can be relaxed so that less reliable feature values will still verify its presence. The current context plus model knowledge determine how much the thresholds should be relaxed.

Although top-down control can be very effective, bottom-up, data-driven control is also necessary. It is essential in providing the initial context that the previously-mentioned top-down method can use. It is also important in scenes where context is lacking.

Once a schema is instantiated from the data by having its slots filled, then two types of messages can be sent. First, the schema will generally transmit a bottom-up request to higher-level schemata to discover where it should fit into the semantic network. For example, a newly instantiated road will send a message to "road system" which looks for a compatible existing road-system instance to join. If one does not exist, it will be created. Second, the schema may send out suggestions of other possible interpretations. While one road schema was being instantiated, an intersecting, bright, linear region would be a candidate for another road. These suggestions could go in any direction in the hierarchy--up, down, or laterally.

Once processing has passed the first few stages, there will generally be several possibilities for further processing. These are placed in a priority queue which can be modified by the user (cf. the next section). These possibilities include a mixture of both top-down and bottom-up messages. Control will move up and down the hierarchies and at each node more messages will be spawned to induce more processing.

This type of processing is neither top-down nor bottom-up. It takes advantage of the relations between schemata which organizes them into a graph (see also [10]). A message sent from a schema may be going up the hierarchy (bottom-up), down it (top-down), or laterally across. The latter mode is similar to heterarchical processing [11]. The strategy is best-first or island-driven with the rankings on the priority queue used to determine the order of search. Since hypotheses can be retracted, it is more akin to non-monotonic reasoning than most vision systems which usually follow the principle of least commitment. It is also described well and is independently motivated by the cycle of perception.

Interaction

The cycle of perception provides convenient points for communication with the user. At each node in the cycle there is a particular, useful kind of information that can be exchanged. These inputs and outputs are shown in Figure 1 and are described below.

Outputs

GLOBAL: When a schema is invoked, the model type (e.g. river) can be described for the user so that he/she knows what the current focus of attention is.

LOCAL: The information derived from the image can be output when the image is sampled. It can be in the form of verifications of hypotheses directed by the schema, facts concerning the image features that may be relevant to the schema, or data that can cause a shift in attention.

PLANNING: This is a list of possibilities for further processing. As they are executing, the schema-specific procedures can make inferences about other models that should be invoked. They will then send a message to alert the appropriate schema. Those messages plus the user's input make up the possibilities list.

The schema-specific procedures will give up control for one of three reasons. The procedure might succeed and the current schema will become part of the semantic network. It might fail, in which case the hypothetical schema will be destroyed. Or it might suspend to wait for more information. Attention would then normally shift to the first entry in the possibilities list.

The possibilities list is a priority queue. Its entries are ranked by the schema that sent the message. The value given by the schema will generally be a factor of its own confidence and the certainty of the inference that caused the message to be sent. The rankings are shown to the user so that he/she can see how processing will continue and can assert his/her priorities by rearranging the queue.

Inputs

GLOBAL: The user can enter general or specific information about the scene. General data includes parameters that affect all levels of processing, for example, the scale of the aerial photograph. Specific information pertains to specific objects. For instance, one can indicate that there is a road in the picture. This information can cause entries to be added to the possibilities list or might cause global schemata to be modified.

LOCAL: More specific information can also be introduced concerning the interpretation that is taking place. This would include advice to the procedures of particular invoked schemata.

PRIORITIES: The user can influence subsequent processing by modifying the priority queue. Rearranging the entries changes the search and instantiation priorities of objects. Objects that are of no interest or that are perceived to be false can be removed from the queue. By altering the parameters within the entries, one can modify the interpretation context. This would include changing the location in the image where the object will be sought.

The cycle of perception is well-suited to a schema-based, object-oriented vision system. It allows feedback from previously instantiated schemata to provide a more informed context for subsequent processing. Also, interaction is facilitated by having clearly distinguished points in the cycle where different types of information can be exchanged.

The Implementation

Figure 3 shows the MISSEE system [12]. Sketch maps are analyzed by Mapsee2 [13] which results in a semantic network of MAYA [14] schemata. The digitized images are analyzed with respect to both edges and regions, yielding primal sketch-like descriptions. A more complete description of the MISSEE system can be found in [15].

Control is exercised through the cycle of perception by means of a global priority queue. Each message on the queue designates a recipient schema, how it is to be entered, and a possible context for the evaluation of one of its attached procedures. This is analogous to, but more limited than, pattern-directed invocation and achieves a similar modularity.

Messages are sent from schemata to redirect attention to other schemata as the results of interpretation become available. The user can also initiate messages either to impart new information or to make known his/her requirements. For modularity, the sender of the message (a schema or the user) only needs to "know" two aspects of the receiver: the name of the recipient schema as well as where it sits in the hierarchy relative to the sender. Lateral messages are always sent top-down. A priority value is included to rank the message. The user can provide a value to reflect the importance of his/her message relative to those already in the queue. A schema will generally use its CONFIDENCE value plus or minus some small number to reflect the importance of the message. Finally, some information useful to the procedure, such as location, may be provided.

Another global facility is the demon list. Demons, functions established by attached procedures, are activated when some future condition becomes true. After every attached procedure is executed, each demon is evaluated if its initiation conditions are true. Normally, after a demon has run it will remove itself from the list.

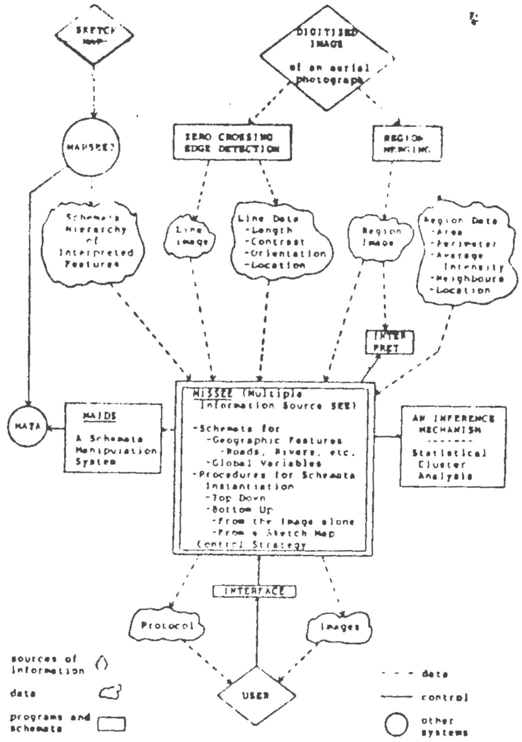

SKETCH MAP
DIGITIZED IMAGE of an aerial photograph
MAPSEE2
ZERO CROSSING EDGE DETECTION
REGION MERGING
Schemata Hierarchy of Interpreted Features
Line image
Line Data -Length -Contrast -Orientation -Location
Region image
Region Data -Area -Perimeter -Average Intensity -Neighbours -Location
INTER PRET
MAYA
MAIDS A Schemata Manipulation System
MISSEE (Multiple Information Source SEE)
-Schemata for
-Geographic Features
-Roads, Rivers, etc.
-Global Variables
-Procedures for Schemata Instantiation
-Top Down
-Bottom Up
-From the Image alone
-From a Sketch Map
Control Strategy
AN INFERENCE MECHANISM
Statistical Cluster Analysis
INTERFACE
Protocol
Images
USER
sources of information
data
programs and schemata
data
control
other systems

Figure 3. The MISSEE System

Demons are used in MISSEE in conjunction with lateral messages. For example, while instantiating a bridge, a message may be sent to suggest a region that might correspond to a river. At the same time, a demon will be set up to wait for the river to actually be instantiated. If it is, the demon will create a neighbour link between the two instances and vanish. The demon's action saves the river schema from having to search all of its neighbouring regions for one corresponding to a bridge instance.

Bottom-up procedures are used to instantiate schemata and to build up the semantic network. Top-down procedures are used to direct attention to schemata whose instantiation would most profitably advance the interpretation.

In MISSEE, the user is able to do as much or as little as he/she desires to influence the progress of interpretation. The user would generally vary the amount of interaction depending on the accuracy of processing, his/her priorities, or both. By taking an active role, the user can guarantee that the results will be to his/her liking. Depending on how well the automatic system is performing, this may take more or less effort.

At one extreme, the user may choose to let the system run automatically. It is not required that he/she draw a sketch map. In the current implementation, he/she must help train the intensity categorizer to assign possible classes to regions in the image, but there are ways to make this automatic, too.

The user can ensure that the end results will be perfect. However, since there is no "natural" user interface, more extreme actions require the user to have more knowledge about the nature of messages and the schemata manipulation functions.

With regard to task priorities, the user can narrow the system's focus. If the user is only interested in objects of a certain type, such as roads or river systems, then he/she can direct MISSEE to look for only those objects (and their components and specializations) and to ignore irrelevant messages. In addition, the user can be very specific about where to find the items of interest. Then, he/she can examine specific parts of the resulting schemata to obtain the desired information about the objects.

Results

The goal of the cycle of perception is to use the results of interpretation to guide further processing so that scene knowledge is accumulated in an intelligent manner. Instead of randomly searching the image for interesting features, the focus of attention shifts in a direction determined by model knowledge and newly acquired information. A summary of the results of applying MISSEE to six images is shown in Table 1. The values indicate the number of regions correctly or incorrectly identified. The overall error rates are 24% when only the image is available and 9% when a sketchmap is also used. This illustrates its effectiveness when only one or two information sources is available.

IMAGE	WITHOUT SKETCHMAP		WITH SKETCHMAP	
	CORRECT	INCORRECT	CORRECT	INCORRECT
Ashcroft	43	22	25	2
Houston	49	13	14	2
Spences Bridge	62	29	15	4
Spences Br. W.	89	11	10	0
Spences Br. E.	18	6	11	1
Cranbrook	13	5	16	0

Table 1. Results of Interpreting Several Images

Search in the image is automatically reduced in three ways. 1) Positional information from the sketch map guides instantiation in the registered intensity image. 2) Model knowledge filters regions for further processing based on the size and average intensity of the region. 3) Model knowledge plus context (the information gained from the interpretation process) suggest likely interpretations for regions near the current focus of attention. Furthermore, possibilities for search are made explicit, giving the user control over their eventual use.

Table 2 shows the number of regions that were searched to instantiate the five bottom level schemata.

As would be expected, the number of regions searched when the sketch map was employed was considerably smaller than without the sketch map. On the average, 22% of the regions were examined when the sketch map was used and 41% when it was not.

IMAGE	TOTAL REGIONS	NUMBER SEARCHED	
		WITHOUT SKETCHMAP	WITH SKETCHMAP
Ashcroft	75	39	23
Houston	139	82	22
Spences Bridge	161	97	35
Spences Bridge West	125	72	32
Spences Bridge East	119	58	45
Cranbrook	285	22	44

Table 2. The Number of Regions Searched

Conclusions

The major hypothesis being tested by this work is that multiple sources of information are "better" than a single source for the interpretation of images. MISSEE is procedurally adequate because it fulfills the two requirements stated in the introduction. Superior results were obtained when the second information source was added (Table 1) and they were obtained more efficiently (Table 2).

By providing several types of attached procedures for each schema, MISSEE ensures that the appropriate method will be used for interpretation. Since the procedures are algorithms, they are flexible, easy to program, and effective at image interpretation. Also, because the procedures are tightly coupled with the declarative portion of the schema greater modularity is maintained.

Schemata cooperate by means of the messages that are passed between them. This is an effective means of shifting the focus of attention to build the semantic network, interpret bottom level schemata, and provide context for subsequent processing.

The control regime, the cycle of perception, is the prime source of procedural adequacy. It allows the user to give and receive information at timely intervals so that he/she can influence processing. Furthermore, it ranks the interpretation possibilities to provide a focus of attention. It fits together with the hierarchies in the semantic network to both fit new knowledge into its proper position as well as using the existing structure to guide subsequent processing.

References

[1] W.S. Havens and A.K. Mackworth, "Representing Knowledge of the Visual World", Computer (Oct., 1983), pp. 90-96.

[2] A.K. Mackworth, "How to See a Simple World", in Machine Intelligence 8, E.W. Elcock and D. Michie (eds.), John Wiley, (1977), pp. 510-540.

[3] J. Glicksman, "A Schemata-Based System for Utilizing Cooperating Knowledge Sources in Computer Vision", In Proc. Fourth Conf. CSCSI, Saskatoon, Canada, (May, 1982), pp. 33-40.

[4] L.D. Erman, F. Hayes-Roth, V.R. Lesser, and D.R. Reddy, "The HEARSAY-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty", Computing Surveys (June, 1980), pp. 213-253.

[5] A.R. Hanson and E.M. Riseman, "VISIONS: A Computer System for Interpreting Scenes", in Computer Vision Systems, A.R. Hanson and E.M. Riseman (eds.), Academic Press (1978), pp. 303-334.

[6] R.A. Brooks, "Symbolic Reasoning Among 3-D Models and 2-D Images", Artificial Intelligence (Aug., 1981), pp. 285-348.

[7] U. Neisser, Cognition and Reality: Principles and Implications of Cognitive Psychology, W.H. Freeman, (1976), 230 pp.

[8] A.K. Mackworth, "Vision Research Strategy: Black Magic, Metaphors, Mechanisms, Miniworlds and Maps", in Computer Vision Systems, A.R. Hanson and E.M. Riseman (eds.), Academic Press, (1978), pp. 53-61.

[9] D.Marr, "Early Processing of Visual Information", Phil. Trans. Royal Society of London (Oct., 1976), pp. 483-524.

[10] A.M. Collins and E.F. Loftus, "A Spreading Activation Theory of Semantic Processing", Psychological Review (June, 1975), pp. 407-428.

[11] P.H. Winston, "The MIT Robot", in Machine Intelligence 7, B. Meltzer and D. Michie (eds.), American Elsevier, (1972), pp. 431-463.

[12] J. Glicksman, "Using Multiple Information Sources in a Computational Vision System", In Proc. IJCAI-83, Karlsruhe, W. Germany, (Aug., 1983), pp. 1078-1080.

[13] W.S. Havens and A.K. Mackworth, "Schemata-Based Understanding of Hand-Drawn Sketch Maps", In Proc. Third Conf. CSCSI, Victoria, Canada, (May, 1980), pp. 172-178.

[14] W.S. Havens, A Procedural Model of Recognition for Machine Perception, Dept. of Computer Science, U.B.C., TR-78-3, (March, 1978), 207 pp.

[15] J. Glicksman, A Cooperative Scheme for Image Understanding Using Multiple Sources of Information, Dept. of Computer Science, U.B.C., TN 82-13, (November, 1982), 286 pp.

THE LOCAL STRUCTURE OF IMAGE DISCONTINUITIES IN ONE DIMENSION

Yvan Leclerc
Steven W. Zucker

Computer Vision and Robotics Laboratory
Department of Electrical Engineering
McGill University

Abstract – We present a new method for locating discontinuities in one-dimensional cuts through idealized images. The model for such a cut is a sampled, piecewise smooth (C^1) function corrupted by noise. The new method uses an extension of the mathematical definition that a function is discontinuous at a point if the left- and right-hand limits of the function (or any of its derivatives) are unequal. The extended definition is that a noisy function is discontinuous at a point if the difference in the estimates of the limits is statistically significant. It follows from this extended definition that locating discontinuities must be carried out at the same time as determining the structure of the uncorrupted function in a local neighbourhood about the discontinuity (we call this the local structure of the discontinuity). Determining this local structure is an important first step in interpreting the physical event causing the image discontinuity.

Introduction

Image intensity discontinuities play a crucial role in the interpretation of images because they can only correspond to one thing – discontinuities in the physical events being viewed. For example, the labeled regions of Figure 1(a) all contain discontinuities in the physical events viewed; be they discontinuities in surface depth, surface orientation, reflectance, or lighting. In a real image, some of these events might be out of focus or some of the cast shadows might have fuzzy boundaries due to non-point sources of light, etc. Such an image would therefore have a combination of image intensity discontinuities (for those events that are in focus), and "image events at larger scales" [1]. To focus our attention on image intensity discontinuities *per se*, we will consider an idealized world in which the surfaces and their reflectance properties are piecewise smooth, with only point sources of light and no diffraction. The image of such a world taken by an idealized camera in which everything is perfectly in focus is a piecewise smooth intensity function.

We tackle two problems with regards to these idealized images. The first is locating the discontinuities in a one-dimensional cut of this piecewise smooth intensity function given a noisy, sampled version of the cut. The second is determining what we call the local structure of these discontinuities. In the following Section we discuss why for determining local structure is important for image interpretation. Following this, we show that locating discontinuities cannot be done without simultaneously determining local structure. Next, we describe a technique for doing both simultaneously and present some results of the technique.

Why Determining Local Structure is Important for Image Interpretation

Several researchers have noted that the structure of the intensities in a local neighbourhood around a discontinuity is a function of the physical events being viewed [2, 3]. We call this the local structure of the discontinuity. In general, the local structure can be quite complex and is often characteristic to the physical event corresponding to the discontinuity. Thus, capturing the local structure can be an important first step in image interpretation. For the special case of one-dimensional cuts of idealized images, the local structure is the left- and right-hand limits of the intensity function and its derivatives.

As an example of the characteristic local structure of the image of a physical event, consider the boundary of the cast shadow in the left-hand CS region of Figure 1(a). Given that the region is planar, and has a smoothly varying Lambertian reflectance, the cast shadow effectively multiplies the image intensity within its boundary by a constant factor. Thus, the cast shadow adds a constant to the logarithm of the image intensity within its boundary. Because derivatives are unaffected by the addition of a constant, the values of all of the derivatives of the logarithm of the intensity just to the left and right of the discontinuity MUST be identical. This example is illustrated in Figures 1(b) and (c). Figure 1(b) is a graph of the logarithm of the intensities along a horizontal cut through the region when there is no cast shadow.

Figure 1(c) is a graph of the same cut when there is a cast shadow. Note in particular that the first derivatives (slopes) just to the left and right of the discontinuity are identical.

The above example is not meant to indicate that cast shadows have identical left- and right-hand limits of derivatives in all circumstances (they do not). Rather, the example is there to show that local structure is often characteristic of particular physical situations (in this case a shadow cast on a planar Lambertian surface), and therefore capturing this local structure is an important first step in the interpretation of images.

Why Determining Local Structure is Necessary for Locating Discontinuities

A second motivation for determining local structure is that it will necessarily affect whatever process is used for the location of discontinuities. In particular, techniques that assume that all discontinuities have the same, fixed, local structure are bound to be in error wherever the assumption is false (an example that illustrates this follows). This is inevitable for at least some discontinuities, even in idealized images of general scenes. For example, every discontinuity in Figure 2(a), which is a typical cut through an idealized image, has a different local structure. Since local structure must be known to properly locate discontinuities, it follows that locating discontinuities must be done concurrently with determining their local structure.

Many current techniques assume that local structure is the same for all discontinuities [1, 4, 5]. The first two, based on "gradient estimation", implicitly assume that every discontinuity is a local step function, as illustrated in Figure 2(b). Both approaches also suffer from a problem in principle: gradients are undefined at points of discontinuity, hence any "estimate" of the gradient at these points is necessarily meaningless. The techniques described by de Souza, based on sliding statistical tests, either assume that the function is piecewise constant, or that the function only differs by an additive constant across the discontinuity (which is correct for certain cast shadows, as described above, but is incorrect for other types of discontinuities).

Why Zero-Crossings Can't Work. To illustrate the kinds of problems that can arise when assuming that local structure is the same for all discontinuities, consider the Marr-Hildreth zero-crossing technique. The basic algorithm is to find the points of inflection (zero-crossings of the second derivative) of a function blurred by a Gaussian. Applying this to the piecewise smooth function of Figure 2(a) produces the zero-crossings of Figure 2(c) and (d), indicated by vertical lines superimposed on the original function.

Two problems with the technique are immediately apparent.

1. Only the discontinuities that are approximately local step functions are correctly located. For the small σ of Figure 2(c), this corresponds to the first and fifth zero-crossings, while for the larger σ of Figure 2(d), this corresponds to just the first zero-crossing.

2. Many of the zero-crossings do not correspond to any discontinuities at all. In particular, the last three in Figure 2(c) and the last one in 2(d) correspond to ordinary points of inflection of the function.

The technique described in [4], while more resistant to noise, has very similar problems.

In short, the inflection points of smoothed functions are not guaranteed to correspond to discontinuities unless the function is originally piecewise constant. This is not the case, even for *idealized* images of general scenes.

Overview of Our Approach

Our approach is based on the application of estimation theory to the mathematical definition of discontinuity: a function is discontinuous at a point if and only if its left- and right-hand limits are unequal. Specifically, we:

- estimate the left- and right-hand limits of the function and of its derivatives at every point;
- determine whether certain necessary conditions for estimation are satisfied;
- if so, determine whether the difference between the limits is statistically significant.

For a neighbourhood size of $2N$ samples, we estimate the left-hand limit of a function at a given point as follows. Define the origin to be midway between two consecutive samples. Estimate the function to the left of the origin using the left-most N samples (we call this the left half-neighbourhood). The value of this estimated function at the origin is the left-hand limit. Similarly, the value of the derivatives of this estimated function at the origin are the left-hand limits of the derivatives. The analogous procedure using the right half-neighbourhood produces the right-hand limits.

The key to this approach is the procedure for explicitly verifying that the necessary conditions for estimation are satisfied at every point. We examine, case by case over a range of neighbourhood sizes, those situations in which these necessary conditions might be violated, and inhibit the estimate whenever the conditions are in fact violated. In addition, multiple neighbourhood sizes allows us to automatically use larger neighbourhoods for estimation (which is more reliable) when the discontinuities are far apart, and smaller neighbourhoods when the discontinuities are closely packed.

The necessary conditions for our estimation procedure are:

C1. Each half-neighbourhood must correspond to a C^1 piece of the function: that is, the half-neighbourhoods must not contain any discontinuities.

C2. The class of estimating functions must be appropriate: that is, the error introduced by restricting the estimation to this class must be small relative to the error introduced by the noise process.

Although this approach can be applied to many types of underlying functions corrupted by many types of noise, we will be presenting an application of the approach to one-dimensional piecewise C^1 functions corrupted by additive white Gaussian noise. There are several reasons for this choice. First, as discussed in the first section, piecewise C^1 functions are idealizations of one-dimensional cuts through images. Second, additive white Gaussian noise allows us to make the presentation simple by keeping the estimation theoretical analysis simple.

Estimating left- and right-hand limits differs from the techniques described previously in two significant ways. First, this approach estimates the local structure at each discontinuity, rather than assuming that it is the same for each discontinuity. Second, the limits that we estimate are defined everywhere, whereas the gradients "estimated" by the other techniques are undefined at discontinuities. Yet this is precisely where the "estimates" of the gradients are used.

Design Issues

Sensitivity and Reliability

Estimation procedures tend to be more reliable (and hence more sensitive) as the number of samples used for estimation increases. Thus, the estimation procedure should use as large a neighbourhood as possible to increase sensitivity. At the same time, the need to locate closely packed high contrast discontinuities suggests that a small neighbourhood size should be used. These apparently contradictory requirements can be met through the use of multiple-sized neighbourhoods. However, this introduces the problem of consistency between multiple results. The following section addreses this problem.

A Constraint on the Domain of Estimation

There are two constraints that estimation theory imposes on our domain. The first is the minimum rate at which the C^1 pieces of the function must be sampled, and the second is the minimum spacing between the discontinuities.

The first constraint comes from the requirement that the number of degrees of freedom for an estimation problem be strictly greater than zero. That is, for the neighbourhood over which the estimation is performed, the underlying function must be defined using fewer parameters than samples. This is clearly not the case for arbitrary C^1 function. However, almost all C^1 functions can be sampled at a rate such that they are approximately first order over a given number of samples everywhere (call this number of samples N_{min}). Thus, demanding that the function be sampled at this rate allows us to use a first order approximation over any N_{min} consecutive samples within the C^1 pieces of the function. Imposing this constraint on the sampling rate satisfies the estimation theoretical requirement without significantly reducing the space of C^1 functions. An important note is that this minimum sampling rate does NOT imply that a first order approximation (or any other specific order of approximation) is appropriate for more than N_{min} consecutive samples. Indeed, one of the problems we face is choosing an appropriate order of approximation for more than N_{min} samples, as discussed later.

The second constraint is that the minimum distance between successive discontinuities be N_{min} samples. This constraint allows us to use a neighbourhood of N_{min} samples on either side of a discontinuity which is guaranteed to be approximately first order without overlapping other discontinuities. As we shall see, this allows us to use a particular technique called intra-scale inhibition to inhibit the estimation procedure in places where the necessary conditions for estimation are not satisfied.

Details of the Procedure

The details and results that follow are for the special domain of piecewise C^1 functions corrupted by additive white Gaussian noise. We use n^{th} order polynomials as our estimating functions to facilitate both the least-squares estimation of the limits and the determination of statistical significance.

Estimating the Limits

We estimate the function in each half-neighbourhood as follows. Define a local coordinate system with the origin mid-way between two consecutive samples. In this way, the coefficients of the polynomials are the limits of the polynomials and their derivatives at the origin. The least-squares approximation of the two independent polynomials is carried out as follows.

Let:

- $\vec{Y}_\ell$ be the column vector of observations for the left half-neighbourhood;
- $\vec{Y}_r$ be the column vector of observations for the right half-neighbourhood;
- $\vec{X}_\ell$ be the column vector of coordinates for the left half-neighbourhood: $(-(2N-1)/2, -(2N-3)/2, \ldots, -3/2, -1/2)$;
- $\vec{X}_r$ be the column vector of coordinates for the right half-neighbourhood: $(1/2, 3/2, \ldots, (2N-3)/2, (2N-1)/2)$;
- $\vec{U}_\ell$ be the column vector of unknown (noise) values for the left half-neighbourhood;
- $\vec{U}_r$ be the column vector of unknown (noise) values for the right half-neighbourhood;
- $\vec{0}$ be a column vector of N zeros;
- $\vec{1}$ be a column vector of N ones;

then the model for the estimate of the two independent n^{th} order polynomials with the origin defined at the center can be written as follows [6]:

$$\begin{bmatrix}\vec{Y}_\ell \\ \vec{Y}_r\end{bmatrix} = \begin{bmatrix}\vec{1} & \vec{0} & \vec{X}_\ell & \vec{0} & \vec{X}^2_\ell & \vec{0} & \cdots & \vec{X}^n_\ell & \vec{0} \\ \vec{0} & \vec{1} & \vec{0} & \vec{X}_r & \vec{0} & \vec{X}^2_r & \cdots & \vec{0} & \vec{X}^n_r\end{bmatrix} \begin{bmatrix}b_{\ell 0} \\ b_{r0} \\ b_{\ell 1} \\ b_{r1} \\ b_{\ell 2} \\ b_{r2} \\ \vdots \\ b_{\ell n} \\ b_{rn}\end{bmatrix} + \begin{bmatrix}\vec{U}_\ell \\ \vec{U}_r\end{bmatrix}$$

or

$$\vec{Y} = X\vec{B} + \vec{U}.$$

The maximum-likelihood vector of parameters $\vec{B}$ can be computed using the matrix product

$$\vec{B} = (X'X)^{-1}X'\vec{Y}.$$

Since the coordinate system is defined locally, the matrix $T = (X'X)^{-1}X'$ is the same for all points. Thus, at each point, a given coefficient of $\vec{B}$ is the inner-product of a *constant* row of T with the sampled data $\vec{Y}$. This is equivalent to convolving the sampled data with the reverse of this row vector, a very efficient computation.

For example, the difference between the left- and right-hand limits of the estimate is $(\hat{b}_{\ell 0} - \hat{b}_{r0})$. This is equivalent to convolving the sampled data with $(T_1 - T_2)$, illustrated in Figure 3.

Verifying the Necessary Conditions

The procedure for verifying that conditions **C1** and **C2** are met is two-fold. The first step of the procedure applies to the smallest half-neighbourhood size N_{min} as follows. Recall that we have constrained our domain such that successive discontinuities are no closer than N_{min} samples apart, and that the C^1 function is approximately first order over N_{min} contiguous samples. Given this constraint, a half-neighbourhood of size N_{min} can contain at most one discontinuity. Also, by choosing 1^{st} order polynomials, condition **C2** is necessarily satisfied and need not be verified. Thus, for a half-neighbourhood size of N_{min}, we are left with three possibilities:

1. there is no discontinuity within the neighbourhood, in which case the estimation procedure is valid;
2. there is one discontinuity that is in the center of the neighbourhood, in which case neither of the half-neighbourhoods contain discontinuities and again the estimation procedure is valid; or
3. there is one discontinuity within either the left or right half-neighbourhood (or both).

See Figure 4 for examples of these three cases. Since the estimates are only valid for cases (1) and (2) above, what needs to be done is to inhibit the estimation procedure in case (3). The observation that allows us to do this is that, all else being equal, the error in estimating the function in case (3) will be considerably larger than in case (2). Thus, by choosing only those estimates that are locally minimum in error (i.e., those estimates that have smaller sum of squared errors than those within a distance of N_{min} samples), we inhibit all estimates that fall into the third category.

Figures 5(a) to 5(d) illustrate what can happen if the verification stage of the estimation procedure is left out. In Figure 5(a), we have applied the estimation procedure at every point for a half-neighbourhood size of $N_{min} = 4$ using first order approximations, and have highlighted all points where the difference between the left- and right-hand limits were considered to be statistically significant with vertical lines. Note that some singularities are surrounded by several highlighted points. In Figure 5(b), we have applied the verification procedure, which we call intra-scale inhibition, eliminating estimates for case (3) situations. Note that each singularity now has only one highlighted point.

The second step in the procedure is applicable to all neighbourhoods whose half-neighbourhood size is greater than N_{min}. By applying the same procedure as for the smaller neighbourhoods, we can eliminate invalid estimates for case (3) situations as before. However, there are two more cases that must also be dealt with:

4. there is more than one discontinuity within one or the other half-neighbourhood; and

5. even if there are no discontinuities within the neighbourhood, there is now no guarantee that a given order of approximation is appropriate for this larger neighbourhood size.

In both cases, the expected value of the normalized sum of squared errors will be larger than the standard deviation of the Gaussian noise. By inhibiting estimates whose sum of squared errors is significantly larger than an independent estimate of the standard deviation, we are left with valid estimates.

We can get an approximately independent estimate of the standard deviation as follows. Divide the left half-neighbourhood into non-overlapping intervals of N_{min} samples. Let K denote the number of such intervals that are valid according to step 1 of this procedure. Let $\hat{S}$ denote the sum over these intervals of the sum of squared errors. Because $\hat{S}$ is formed using only *valid* estimates, $\hat{S}/(K(N_{min} - 2))$ is an unbiased estimator of the standard deviation. Let $\hat{S}_\ell$ denote the sum of squared errors over the larger half-neighbourhood. $\hat{S}$ and $\hat{S}_\ell$ are two chi-squared random variables that are approximately independent. Thus, if the order of the polynomial used at the larger neighbourhood size is n, then the statistic

$$\frac{\hat{S}_\ell/(N-(n+1))}{\hat{S}/(K(N_{min}-2))}$$

is approximately $F_{(N-(n+1),K(N_{min}-2))}$ distributed. Thus, we can reject the hypothesis that the sum of squared errors of the smaller and larger neighbourhoods are equal if the statistic is larger than the 0.99 confidence level of this distribution. The analogous test is done for the right half-neighbourhood.

Figure 5(c) illustrates what can happen if step 2 of the procedure is omitted. In this figure, we have applied the estimation procedure at every point for a half-neighbourhood size of $N = 16$ using second order approximations. We have inhibited all estimates using intra-scale inhibition, and have highlighted all points where the difference between the left- and right-hand limits were considered to be statistically significant. Note that the third singularity is not properly identified, and that a point is highlighted where the function is clearly smooth. In Figure 5(d) we have applied the second step of the verification procedure. Note that all singularities separated by at least 16 samples are now highlighted, and that only singularities are highlighted.

Determining Statistical Significance

Having computed the maximum-likelihood vector of polynomial coefficients $\hat{B}$, we can determine whether the coefficients are significantly different from each other. In particular, we are interested in seeing if the 0^{th} and 1^{st} coefficients are significantly different from each other. This is done by setting up three hypotheses, and testing them in turn: they are:

$$\begin{aligned} H_1&: \quad b_{\ell 0} = b_{r0} \quad \text{and} \quad b_{\ell 1} = b_{r1}; \\ H_2&: \quad b_{\ell 0} = b_{r0}; \\ H_3&: \quad b_{\ell 1} = b_{r1}. \end{aligned}$$

If H_1 is accepted, then both coefficients are equal and no further testing is required. If H_1 is rejected, we go on to H_2. If H_2 is accepted then the 0^{th} coefficients are the same, and, as a consequence of the rejection of H_1, the 1^{st} coefficients must be different. If H_2

is rejected, we go on to H_3. If H_3 is accepted, then the 1^{st} coefficients must be the same, and again, as a consequence of the rejection of H_1, the 0^{th} coefficients must be different. Finally, if H_3 is rejected, then both coefficients must be different.

Hypothesis H_1 can be tested by performing a least-squares fit to the constrained model [6]:

$$\begin{bmatrix} \vec{Y}_\ell \\ \vec{Y}_r \end{bmatrix} = \begin{bmatrix} \vec{1} & \vec{X}_\ell & \vec{X}_\ell^2 & \vec{0} & \cdots & \vec{X}_\ell^n & \vec{0} \\ \vec{1} & \vec{X}_r & \vec{0} & \vec{X}_r^2 & \cdots & \vec{0} & \vec{X}_r^n \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_{\ell 2} \\ b_{r2} \\ \vdots \\ b_{\ell n} \\ b_{rn} \end{bmatrix} + \begin{bmatrix} \vec{U}_\ell \\ \vec{U}_r \end{bmatrix}$$

Note that in this constrained model, there is one common coefficient b_0 and one common coefficient b_1, reflecting the hypothesis that both $b_{\ell 0} = b_{r0}$ and $b_{\ell 1} = b_{r1}$. If we denote the sum of squared errors resulting from the least-squares fit of this constrained model as $\hat{S}_1$, then the statistic

$$\frac{(\hat{S}_1 - \hat{S})/2}{\hat{S}/(2N - 2(n+1))}$$

follows the $F_{(2,2N-2(n+1))}$ distribution. Thus, we reject H_1 if the statistic is larger than the .99 confidence level of this distribution.

Similarly, hypotheses H_2 and H_3 can be tested using constrained models reflecting the appropriate equality in coefficients.

Results

We have demonstrated the results of the procedure on a piecewise smooth function with no added noise in Figure 5. In Figure 6 we illustrate the results of the procedure on the same function with added white Gaussian noise of standard deviation 2. Note that very few false discontinuities were found by the procedure, and that most discontinuities that are visible to the eye were properly identified.

Summary and Conclusions

The local structure of image intensity discontinuities can be used to great advantage in the interpretation of images. Yet, the techniques that have been advocated by other researchers for finding discontinuities cannot recover such structure. Indeed, they assume a predetermined structure, and this produces inadequate results when applied to more general domains. A new approach is therefore necessary, and we have developed a technique for doing precisely this in the reduced problem space of piecewise C^1 one-dimensional signals. Examples have demonstrated the power of the technique, even in the presence of a considerable amount of noise.

The general approach we used in developing the technique came directly from the mathematical definition of discontinuity: a function is discontinuous at a point if and only if its left- and right-hand limits do not equal each other. We showed that the key to estimating these limits is verifying that the necessary conditions for estimation are satisfied (this must be done for every estimation problem, another shortcoming of existing approaches). We resolved the verification problem by examining, case by case, those situations in which the necessary conditions would be violated. Thus, we were able to inhibit estimation in these situations through the interaction of certain error procedures over multiple-sized neighbourhoods. In addition, multiple-sized neighbourhoods allowed for the use of many samples for estimation (when appropriate) to increase the reliability of the results, while at the same time accomodating closely packed discontinuities. This produced an algorithm which was very sensitive to low contrast discontinuities when little or no noise was present, and yet rarely produced false discontinuities in the presence of large amounts of noise.

References

[1] D. Marr and E. Hildreth, Theory of Edge Detection, MIT AI Memo 518, April, 1979.

[2] B.K.P. Horn, "Understanding Image Intensities", Artificial Intelligence, (**8**,1977), pp. 201-231.

[3] A.W. Witkin, "Intensity-based Edge Classification", Proceedings AAAI-82, 1982.

[4] R.M. Haralick, "Digital Step Edges from Zero Crossings of Second Directional Derivatives", PAMI, (**1**, 1984), pp. 58-68.

[5] P. de Souza, "Edge Detection Using Sliding Statistical Tests", Computer Graphics and Image Processing, (**23**, 1983), pp. 1-14.

[6] J. Johnston, Econometric Methods, McGraw-Hill, (1963).

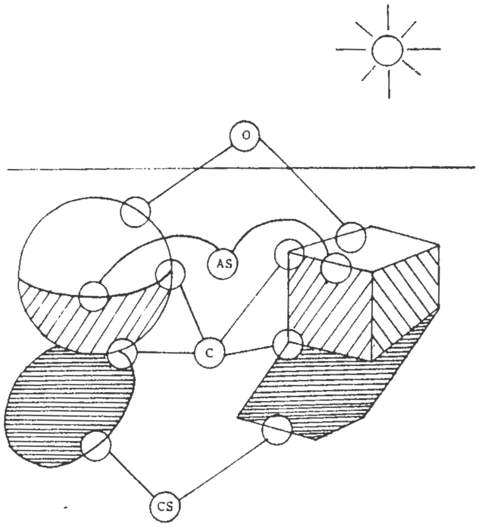


Figure 1. A simple scene illustrating some of the types of image-forming processes that result in image intensity discontinuities. Regions CS contain boundaries of cast shadows; regions AS contain boundaries of attached shadows; regions O contain occlusion boundaries; and regions C contain complex combinations of image-forming processes.

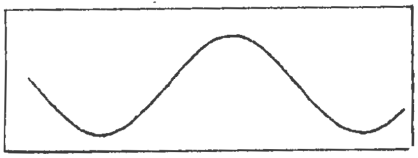


Figure 1(b) Intensity profile without cast shadow.

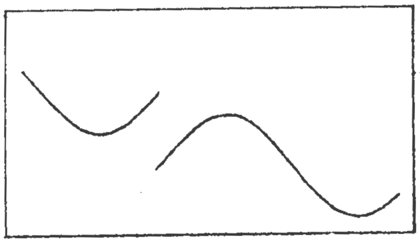


Figure 1(c) Intensity profile with cast shadow.

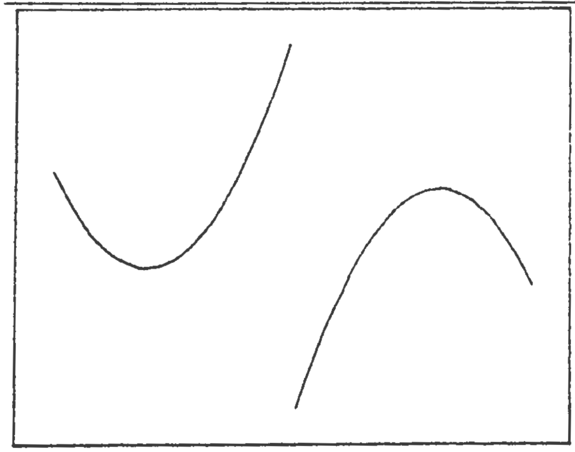


Figure 3. The convolution kernel for estimating the difference in coefficients $b_{\ell 0} - b_{r0}$.

[redacted]

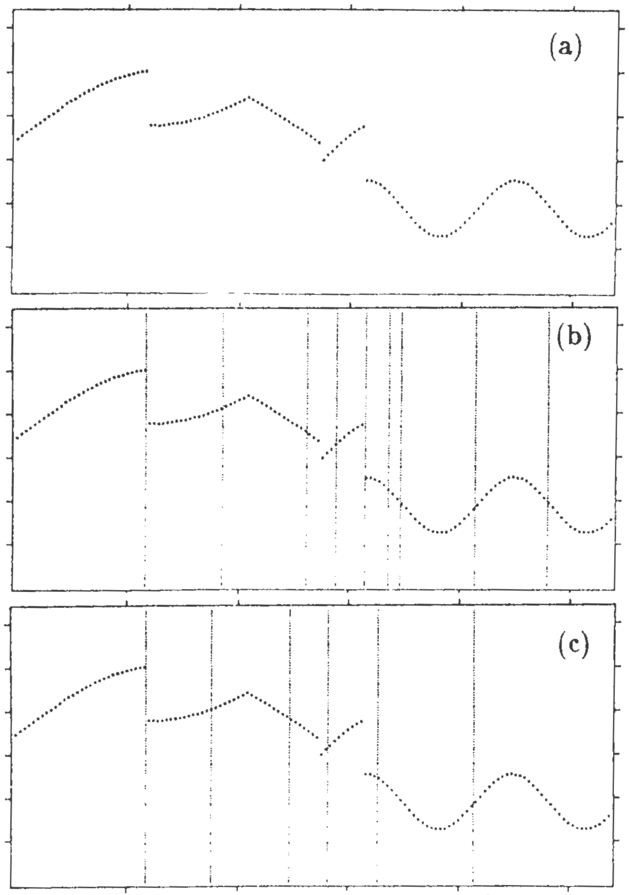


Figure 2. **(a)** A sampled piecewise C^1 function. **(b)** The zero-crossings of the Laplacian of the Gaussian blurred function for $\sigma = 2.5$ (central positive region of about 5 samples). Note that only the first and fifth zero-crossings correspond to discontinuities. **(c)** Same as (b), with $\sigma = 5$ (central positive region of about 10 samples). Note that only the first zero-crossing corresponds to a discontinuity.

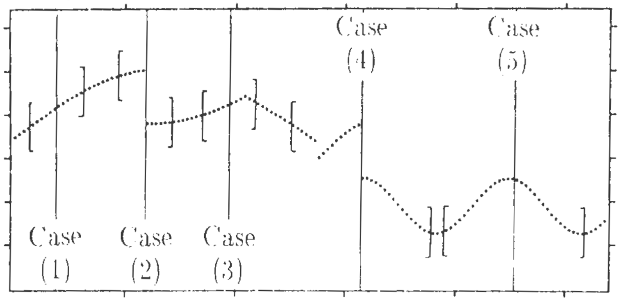


Figure 4. An illustration of the various neighbourhood cases. Cases (1) and (2) provide valid estimates, whereas the others do not, and hence the estimates must be inhibited.

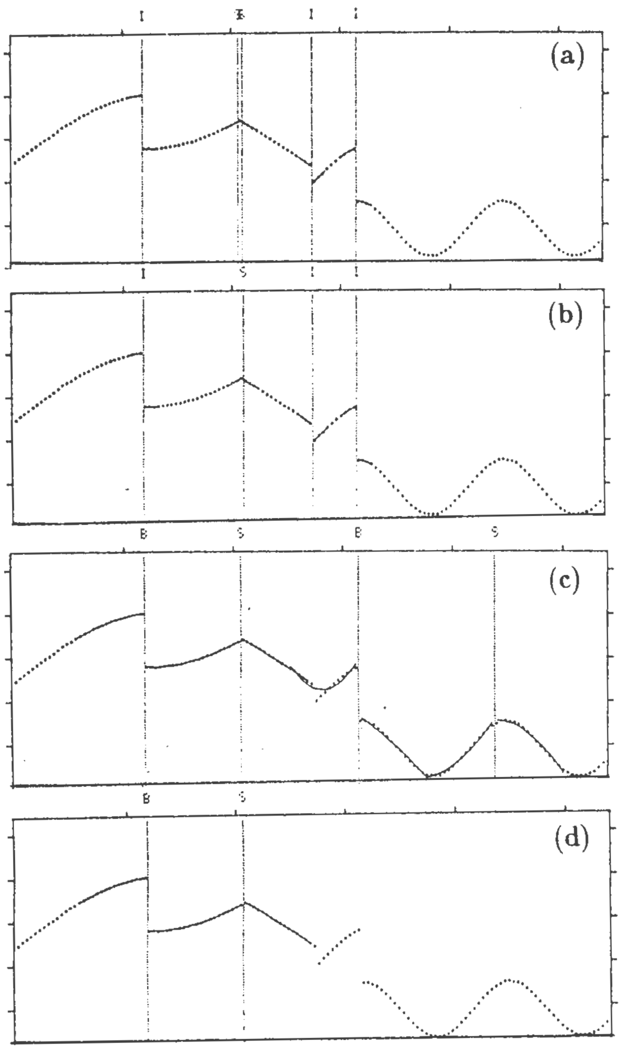


Figure 5. An illustration of the need for the various estimate inhibition procedures. The function is the same as in Figure 2. **(a)** Results of first order approximations over the smallest half neighbourhood size N_{min} = 1 without the inhibition procedure. Note that some discontinuities are surrouned be several highlighted points. **(b)** Same as (a) with intra-scale inhibition applied. Note that now each discontinuity is properly identified. **(c)** Results of second order approximations over a half neighbourhood size of $N = 16$ with intra-scale inhibition, but without inter-scale inhibition. Note that the neighbourhood about the third highlighted point is case (4) and the next is case (5), and therefore cannot be properly dealt with by intra-scale inhibition alone. **(d)** Same as (c) with the addition of inter-scale inhibition. Note that each discontinuity that is separated by at least 16 samples is now properly identified.

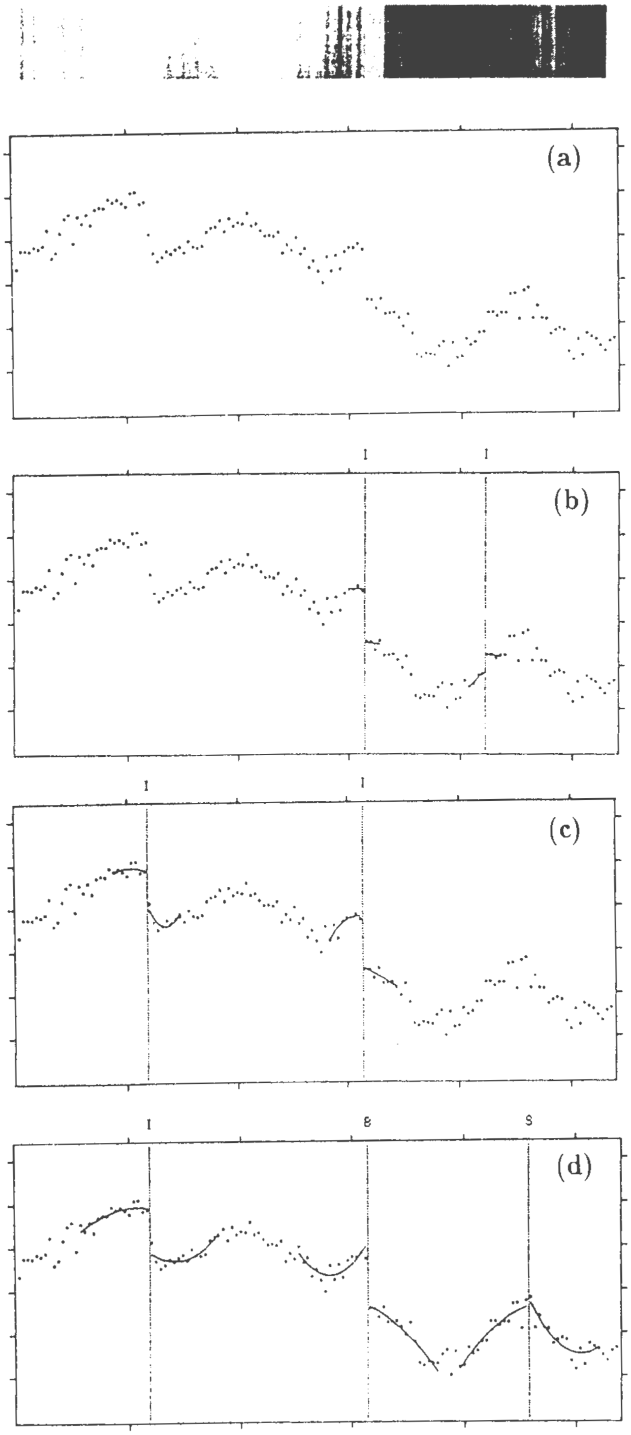


Figure 6. Results using all estimate inhibition procedures on the sampled function of Figure 2 with added white Gaussian noise of standard deviation 2. **(a)** The function with added noise. **(b)** Result of the procedure for $N = N_{min} =$ 1. **(c)** Result of the procedure for $N = 8$. **(d)** Result of the procedure for $N = 16$.

RECEPTIVE FIELDS AND THE RECONSTRUCTION OF VISUAL INFORMATION

Steven W. Zucker
Computer Vision and Robotics Laboratory
Department of Electrical Engineering
McGill University
Montreal, Quebec, Canada

Robert A. Hummel
Courant Institute of Mathematical Sciences
New York University
New York, N.Y.

ABSTRACT

Receptive fields in the retina indicate the first measurements taken over the (discrete) visual image. Why are they circular surround with an excitatory/inhibitory structure? We hypothesize that they provide a representation of the visual information in a form suitable for transmission over the optic nerve, a rather limited channel. The hypothesis is supported by a formal scheme for reconstructing visual information at the cortex. The scheme is both physiologically plausible, and leads to a number of predictions about receptive field size, structure and hyperacuity that are supported. The existence of such a scheme suggests that the analysis of visual information really begins in the cortex, a suggestion that stands in strong opposition to many current beliefs about "edge detection".

We have developed a theory of image reconstruction which explains precisely how it is possible that detailed visual information can be made available to the cortex. The theory provides a single, consistent role for much of the spatial processing along the X-pathway from the retina through the lateral geniculate nucleus to the visual cortex. The need for this information follows, in principle, from the precision with which we can interact with our visual environment [1]. Under the assumption that receptive fields carry measurements that can be represented by operator convolutions, the theory accounts for the following physiological observations: (i) retinal receptive fields have a center surround organization, with an excitatory center and inhibitory surround; as well as (ii) with an inhibitory center and excitatory surround [2]. We are using the terms as defined in [3], to stress the antagonistic manner in which information is combined within a receptive field. (iii) At the cortex, receptive fields span a range of sizes [4]. The theory takes into account that (iv) neurons have a limited capacity to carry information, and that (v) neurons along the X-pathway have a rather low spontaneous firing rate [5]. Finally, for truly accurate reconstructions the theory requires (vi) the presence of additional side lobes in certain cortical receptive fields [6;7]. Since these facts summarize most of what is known about the basic X-pathway, the suggestion that the analysis of visual information begins in the cortex becomes very much more plausible.

The reconstruction scheme is based on a restoration of data obtained from receptive field measurements. An essential feature of these measurements is the blurring or diffusion of information, which we shall model according to the heat equation. While the formal role of the heat equation will be introduced shortly, the intuition comes from the observation that a unit impulse of heat diffuses into increasingly larger Gaussian distributions as time proceeds (Fig. 1). Such a (temporal) spread will become analgous to the (spatial) extent of a receptive field.

Mathematically, let f(x) denote the initial temperature distribution as a function of the spatial variable $x \varepsilon \mathbb{R}^n$. Then a solution to the heat equation $u(x,t)$, $t \geq 0$, satisfying

$$u(x,0) = f(x)$$

can be obtained from the convolution

$$u(x,t) = \int_{\mathbb{R}^n} K(x-x',t)\ f(x')\ dx'$$

where K(x,t) is the "source" kernel [8]

$$K(x,t) = (2\pi t)^{-n/2}\ e^{-|x|^2/4t}.$$

Since the kernel acts as a blurring operator, we can regard the distributions u(x,t) as representing continuously coarser representations of the original data f(x) as t increases. In fact, assuming f(x) is bounded, u(x,t) as given above is analytic. It is the unique bounded solution to the heat equation

$$u_t = \Delta u$$

satisfying u(x,0) = f(x). Other (bounded) solutions are technically possible, but the function u(x,t) given by convolution against the Gaussian kernel K is the one that naturally occurs in physical systems.

Suppose initial image data were blurred by a Gaussian kernel. Is it possible to reproduce the original data? Specifically, given $g(x) = u(x,\tau)$, for some fixed $\tau > 0$, is it possible to solve the heat equation backwards to recover u(x,t) for $0 < t < \tau$?

There are two separate aspects to the answer: whether recovery is possible in principle and whether it is possible in practice. In principle it can be shown that necessary and sufficient conditions for the existence of a solution to the heat equation, u(x,t), $0 \leq t \leq \tau$, satisfying $u(x,\tau) = g(x)$, $x \varepsilon \mathbb{R}^n$, are that g(x) be analytic, and that the extension of g(x) to an analytic function of several complex variables g(z), $z \varepsilon C^n$, satisfies certain growth conditions. Both of these conditions, analyticity are bounded growth, fit smoothly into the vision context.

Given the existence of a backsolution, calculating it may still be impractical. The difficulty is that the backward heat problem is unstable; that is, a small change in the initial data $g(x)$ can lead to an arbitrarily large change in $u(x,t)$. No matter how well $g(x)$ is approximated numerically, there will always exist examples where the resulting calculation for $f(x)$ is arbitrarily far off.

Nevertheless, John [9] has shown that, if a non-negative backsolution exists, then stable reconstruction of $u(x,t)$ is possible for $a \leq t \leq \tau$, where $a > 0$ is fixed. Specifically, suppose that $0 \leq g(x) \leq \mu$, and that $g(x)$ is sampled so that it can be reconstructed to within accuracy ε. Then an estimate $u(x,t)$ can be formed using discrete kernels such that the error $\overline{u}(x,t) - u(x,t)$, for $a \leq t \leq \tau$, is bounded by a constant (depending on a) times $\mu^{1-\Theta} \varepsilon^{\Theta}$. Here Θ is a number between 0 and 1, which implies that, e.g., a twofold increase in the accuracy of reproducing $u(x,t)$ requires more than a twofold increase in the representational accuracy. The coefficients of this kernel are shown in Fig. 2.

In summary, then, the essential restrictions amount to requiring that $g(x)$ be bounded, that a non-negative back solution exists, and that a solution is sought only back as far as some positive time $t \geq a \geq 0$. And, for numerical stability, we must represent $g(x)$ as accurately as possible.

The physiological interpretation begins with the earlier points (i) and (ii): that receptive fields have an antagonistic center/surround organization. Such local operations are useful in data coding, since they compress the required dynamic range of a channel [10,11]. They may also exist in primate visual systems for evolutionary reasons. But, since they are modifying the initial light measurements, they would seem to make reconstruction more difficult. Somehow their effects need to be undone. However, as we now show, given the form of these operators, we just need to add an extra step onto the reconstruction scheme.

Circular surround receptive fields have been modeled by kernels given either as the difference of two Gaussians [12, 13], or as the Laplacian of a Gaussian [14]. Although these kernels are distinct, we can interpret the former as a discrete analog of the latter. This follows since the heat kernel $K(x,t)$, a solution of the heat equation, satisfies

$$\Delta K(x,t) = \frac{\partial}{\partial t} K(x,t) \approx [K(x,t_1)-K(x,t_2)]/(t_1 - t_2)$$

which is the difference of two Gaussians. We therefore take

$$v(x,t) = \int_{\mathbb{R}^n} \Delta K(x - x', t)\ f(x')\ dx'$$

as a continuous version of the initial measurements, where t parameterizes the effective size of the receptive field. Note that $v(x,t)$ is only available for $t \geq \tau$, where τ corresponds to the size of the smallest receptive field. Since

$$v(x,t) = \Delta \int_{\mathbb{R}^n} K(x - x', t)\ f(x')\ dx'$$

$$= \int K(x - x', t)\ (\Delta f)\ (x')\ dx'$$

$v(x,t)$ can be interpreted either as the Laplacian of the blurred intensity data, i.e., as $\Delta u(x,t)$, or as the bounded solution to the heat equation using the initial data $\Delta f(x)$. From the former interpretation and the fact that $u(x,t)$ satisfies the heat equation, we have that $v(x,t) = \Delta u(x,t) = \frac{\partial}{\partial t} u(x,t)$, so

$$-\int_0^T v(x,t)dt = u(x,0) - u(x,T).$$

Now, $u(x,T)$ is nearly constant for large T, so the above integral can be used to recover $f(x)$ modulo an additive constant. From the other interpretation of $v(x,t)$, we see that $v(x,t)$ is itself a solution to the heat equation. Thus the values of $v(x,t)$, for $0 \leq t \leq \tau$, will have to be obtained by backsolving the heat equation using $v(x,\tau)$ as initial data.

To discuss physiological realizations of these formulas, we must confront problems of discretization and stability. The receptive field measurements $v(x,t)$ are not continuous in t, but rather are given by the discrete approximation $t = t_k, t_{k+1}, \ldots, t_m$, with each $t_i \geq \tau$. The data are sampled spatially, and it is likely that $v(x,t)$ is sampled more coarsely in x for larger t, reflecting the more uniform variations of the smoothed data [15]. Values for $v(x,t_i)$, $t = t_1, t_2, \ldots, t_{k-1}$, with $0 < t_i < \tau$, will be obtained by backsolving. An examination of John's coefficients for the backsolution kernel reveals the addition of decreasing side lobes for higher orders of approximation, as would be expected from the de-blurring of Gaussians [16]. The third order approximation agrees nicely with physiological observation (vi); see [6, fig. 9b]. The integral for recovering $f(x)$ can be approximated by a weighted sum

$$-\int_0 v(x,t)\ dt = \sum_{i=1}^{m} (t_i - t_{i-1})\ v(x,t_i)$$

thereby requiring the different size operators (physiological observation (iii)). It is especially interesting to note, with regard to this sum, that it is only the data for t near 0 that needs to be reconstructed, and it is precisely these smaller operators (receptive fields) that have been observed to have the extra side lobes [6]; see also [17] for psychophysical support.

The last remaining issue is stability. John's results require that the backsolution be non-negative, but $v(x,t)$ can be both positive and negative. Therefore we shall split the manner in which $v(x,t)$ is represented, separating the positive and negative parts. Recalling physiological observation (v), that X-pathway neurons have a low spontaneous firing rate, we introduce a smooth approximation to the "positive part" function:

$$\phi_+(x) = \max [x, 0]$$

for $x < A$, a saturation level; see Fig. 3. Similarly, set $\phi_-(x) = \phi_+(-x)$, and note that

$$\phi_+(x) - \phi_-(x) \approx x \qquad \text{for } |x| < A$$

since ϕ is linear in the range in which it is used. We can then separate the receptive-field convolution data into

$$v_+(x,t_i) = \int K(x-x', t_i)\ \phi_+(\Delta f(x'))\ dx'$$

and

$$v_-(x,t_i) = K(x-x',t_i)\ \phi_-(\Delta f(x'))\ dx'$$

Note that this functional form implies that it is only the small convolutions that are sent back from the retina, which is in agreement with the physiology [5; 18]. This formulation also suggests that there should only be a limited range over which X-cells behave linearly, which is also well known. Patterns qualitatively matching these receptive fields are, it is further interesting to note, the ones to which our visual systems are the most sensitive [19]. Since v_+ and v_- are both solutions to the heat equation with (essentially) non-negative data, both can be backsolved from $t = \tau$ to $t = t_1$, with $t_1 > 0$, in a stable fashion. The desired data $v(x,t_i)$, as used before, can then be recovered from the difference

$$v(x,t) = v_+(x,t) - v_-(x,t).$$

In summary, the reconstruction method involves sampling and transmitting the receptive field convolutions $v_+(x,t_i)$ and $v_-(x,t_i)$ for small $t_i \geq \tau$, backsolution of $v_+(x,t_i)$ and $v_-(x,t_i)$ for $t_i < \tau$, and (weighted) summing of all these measurements with the larger, smoothed convolutions. Note that these larger convolutions are obtained just by Gaussian blurring of small center surround receptive fields. The backsolution is stable for both v_+ and v_- provided reconstruction is attempted only as far as some resolution level $t > 0$, as would be expected from hyperacuity data.

Our proposal differs fundamentally from those that implicate the different size operators with notions of "edge detection" [14]. Qualitatively, this other approach asserts that physical events such as reflectance, depth, or illumination changes take place over different spatial "scales", so that different size operators are necessary to capture them. However, this approach suffers from several problems. First, from photometric observations, it can be shown that the physical processes responsible for generating the intensity changes operate over many scales simultaneously [20] and non-linearly. Second, there is the problem of how to combine the information at the different scales. Finally, there is increasingly more psychophysical and computational evidence that early vision in general, and anything like edge detection in particular, requires very precise information [1; 21; 22]. Nevertheless, attempts have been made within this approach to use the zero-crossings of the different size operators as "edge" locations, and diffusion equations have been used to study their migration across scales [20; 23].

While our method has some qualitative similarity to others based on the Shannon sampling theorem (e.g., [24], and references cited therein), there are fundamental differences. The spatial support required for sin x/x reconstruction is larger than the local polynomials that we incorporated, raising serious questions about accuracy [25]. Also, the idea of backsolving to obtain the highest frequency data is not there. Finally, one of the principle advantages of that scheme - - noise averaging in the larger receptive fields -- is present in our scheme as well.

While our reconstruction method demonstrates the possibility of precise image reconstruction in principle, it does not imply that it is necessarily taking place in practice. Perhaps only part of the scheme is utilized, such as just step (1) above, since this also amounts to an effective Gaussian de-blurring strategy. Many sources of such blur exist early in vision, from receptive field convolutions to motion smear [26] to physiological variation in neuronal conduction velocities. Or perhaps the reconstruction takes place only implicitly, within a subsequent level of processing such as orientation selection [21]. Ultimately, the physiology will decide.

REFERENCES

[1] Barlow, H., Reconstructing the visual image in space and time, Nature, 1979, 279, 189-190.

[2] Kuffler, S., and Nichols, J., From Neuron to Brain, Sunderland, Ma: Sinauer Associates, Inc., 1976.

[3] Stein, A., Mullikin, W., and Stevens, J., The spatiotemporal building blocks of X-, Y-, and W-ganglion cell receptive fields of the cat's retina.

[4] Hubel, D., and Wiesel, T., Functional architecture of macaque monkey visual cortex, PROC. ROY. SOC. (LONDON), B, 1977, 198, 1-59.

[5] Stone, J., Dreher, B., and Leventhal, A., Hierarchical and parallel mechanisms in the organization of visual cortex, Brain Res. Rev., 1979, 1, 345-394.

[6] Movshon, J., Thompson, I., and Tolhurst, D., Spatial summation in the receptive fields of simple cells in the cat's striate cortex, J. Physiol. (London), 1978, 283, 53-77.

[7] DeValois, Albrecht, D., Thorell, L., Cortical cells: Bar and edge detectors or spatial frequency filters, in Frontiers in Visual Science, S. Cool and E. Smith (eds.), New York, Springer, 1978.

[8] Widder, D., The Heat Equation, New York, Academic Press, 1975.

[9] John, F., Numerical solution of the equation of heat conduction for preceeding times, Annali de Matematica, 1955, 17, 129-142.

[10] Barlow, H., Critical limiting factors in the design of the eye and the visual cortex, Proc. Roy. Soc. Lond., 1981, B 212, 1-34.

[11] Srinivasan, M., Laughlin, S., and Dubs, A., Predictive coding: A fresh view of inhibition in the retina, Proc. Roy. Soc. Lond., 1982, B, 427-459.

[12] Rodieck, R., Quantitative analysis of cat retinal ganglion cell response to visual stimuli, Vis. Res., 1965, 5, 583-601.

[13] Enroth-Cugell, C., and Robson, J., The contrast sensitivity of retinal ganglion cells of the cat, J. Physiol. (Lond.), 1966, 187, 517-552.

[14] Marr, D., and Hildreth, E., Theory of edge detection, Proc. Roy. Soc. (London), B, 1980, 207, 187-217.

[15] Burt, P., and Adelson, E., The laplacian pyramid as a compact image code, IEEE Trans. Comm., 1983, COM-31, 532-540.

[16] Kimia, B., and Zucker, S.W., Deblurring gaussian blur, IEEE Conf. Computer Vision and Image Proc., Washington, D.C., June, 1983.

[17] Wilson, H., McFarlane, D., and Phillips, G., Spatial frequency tuning of orientation selective units estimated by oblique masking, Vis. Res., 1983, 9, 873-882.

[18] Schiller, P., Finlay, B., and Volman, S., Quantative studies of single-cell properties of monkey straite cortex. I. Spatiotemporal organization of receptive fields, J. Neurophys., 1976, 6, 1288-1319.

[19] Watson, A., Barlow, H., and Robson, J., What does the eye see best?, Nature, 1983, 302, 419-422.

[20] Witkin, A., Scale space filtering, Proc. 8th Int..Joint Conf. on Artificial Intelligence, Karlsruhe, W. Germany, 1983, 1019-1022.

[21] Zucker, S.W., Cooperative grouping and early orientation selection, in O. Braddick and A. Sleigh (eds.), Physical and Biological Processing of Images, Springer, New York, 1983.

[22] Leclerc, Y., Ph.D. Thesis, McGill University, in preparation.

[23] Yuille, A., and Poggio, T., Scaling theorems for zero-crossings, A.I. Memo 722, Cambridge: M.I.T.; 1983.

[24] Sakitt, B., and Barlow, H., A model for the economical encoding of the visual image in cerebral cortex, Biol. Cybern., 1982, 43, 97-108.

[25] Hummel, R., Sampling for spline reconstruction, SIAM J. Appl. Math., 1983, 43, 278-288.

[26] Burr, D., Motion smear, Nature, 1980, 284, 164-165.

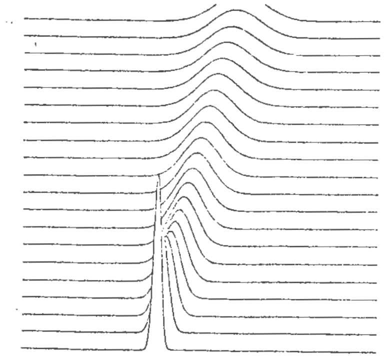


Fig.1. The Gaussian kernel for increasing time; note how, as time proceeds, the kernel spreads out, so that time here becomes analogous with the spatial coefficient (or "scale") of the Gaussian.

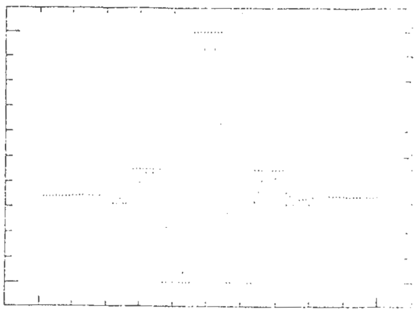


Fig. 2. A comparison of [illegible] necessary [illegible] physiologically

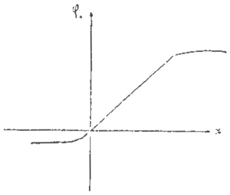


Fig. 3. A sketch of the "positive part" function used to enable neural [illegible] the linear range between the (relatively) [illegible] positive and the [illegible] negative saturation levels.

Trajectory Planning Problems, I: Determining Velocity Along a Fixed Path

Kamal Kant
Steven W. Zucker

Computer Vision and Robotics Laboratory
Department of Electrical Engineering
McGill University

Abstract – In this paper, we generalize the path planning problem to that of the trajectory planning problem **(TPP)** in a time-varying environment. We present an algorithm to solve a special case of the TPP – the **fixed path TPP**. In this case, the path is a point set fixed in space and the aim is to determine the velocity of a robot vehicle moving along this path from an initial position to a final position while avoiding moving obstacles. The solution is essentially equivalent to a (static) 2-D path planning problem. Graph searching techniques are used to construct the velocity profile.

Introduction

The search for a path connecting an initial point with a final point while avoiding obstacles is known as the path planning problem (PPP). The structure of a path planning problem is dependent on apriori constraints on the path and on the environment. Because of the complexity of the general problem [Reif 79, Schwartz 82], researchers have concentrated on subproblems in which different aspects of the environment are highly constrained. For example, many [Perez 79, Brooks 82] have concentrated on the problem of finding paths through static, 2-dimensional worlds, and have discovered algorithms that are global and efficient in these worlds.

In this paper, we approach the PPP from the other direction – that of planning trajectories through 3-dimensional, time-varying environments. We call this the trajectory planning problem (TPP). This perspective (of time varying environments) on the PPP suggests different sub-problems than those that have already been addressed. We concentrate on a special case of the TPP, called the **fixed path TPP**, in which the path of the moving object as a space curve is fixed and the trajectories of the moving obstacles are known. The problem, then, is to find the velocity of the moving object as a function of time along this curve, so that no collisions occur with the moving obstacles. We transform the **fixed path TPP** into an equivalent problem of planning paths in a 2-D static world, the **static 2-D PPP** and solve it with a graph search similar to that in the 2-D PPP.

The Trajectory Planning Problem (TPP)

The TPP is formulated as follows:

Given :

1. A moving object (robot), and
2. Obstacles moving along trajectories (as given functions of time) are completely known.

Aim :

To find the trajectory of the robot as a mapping $f(t)$ from the time interval $[t_i, t_f]$ to E^3; i.e.,

$$f(t) : [t_i, t_f] \rightarrow x \in E^3,$$

where E^3 denotes the 3-D Euclidean space.

Note the difference between the (static) PPP case and the (time varying) TPP case. In the former, the aim is to find the path as a point set x in E^3; in the latter, we are interested in the mapping $f(t)$ as well as its range x. Intuitively, in the static case, once the path has been found, the robot's velocity along it does not matter for collisions. In the time-varying environment, however, it does matter: of two different velocity functions on the same path, one may lead to a collision and the other may not.

The Fixed Path TPP

The above definition suggests a natural special case of the TPP:

> Let the path be given as a space curve, x, parameterized by its arc length s. Find the mapping $f(t)$ from $[t_i, t_f] \rightarrow x(s)$ such that the robot does not collide with the obstacles.

One may view this as follows:

Given that the robot is constrained to move along a fixed path (e.g., it moves on rails which predetermine the path), find the velocity of the robot as a function of time such that collisions with the moving obstacles are avoided. We call this special case the **fixed path TPP**.

We now solve an idealized version of this problem in which the robot is constrained to be a point object moving along the given space curve x.

A Transformation:

From Fixed Path TPP to 2-D Static PPP

An object moving in space sweeps out an effective hyper-volume in space-time. An intuitive solution to the fixed path TPP can be obtained by intersecting the swept volumes of the obstacles with the given path of the robot. These intersections will provide time varying constraints for the robot's position on the path.

Let $s = [s_i, s_f]$ be the segment of the curve on which the robot is constrained to move, with $s(t_i) = s_i$, and $s(t_f) = s_f$. The trajectories of the obstacles are given as functions of time. To compute the intersection of the path $x(s)$ with the volumes swept by the obstacles as they move around is a non trivial task. Since, in general, these intersections will be complicated functions of time depending on:

1. the shape of the obstacle,

2. the trajectory of the obstacle, and

3. the space curve x, i.e., the path of the robot.

As an approximation, these intersections may be taken as a subsegment of the path being unavailable during a subinterval of time. Let t_k be the time instant when obstacle k touches the path segment, and t'_k be the time instant when it leaves the segment. During the whole interval $[t_k, t'_k]$, let the maximum subsegment of the fixed path $x(s)$ occupied by the obstacle k be given by $[s_k, s'_k]$. Once these intersections are available, they act as constraints on the mapping $f(t)$. In other words, $f(t)$ may not map the time subinterval $[t_k, t'_k]$ to the spatial subsegment $[s_k, s'_k]$. This is illustrated in figure 1(a).

Now consider the s-t plane. Since our approximations to the intersections are of the form:

> subsegment $[s_k, s'_k]$ is unavailable during the subinterval $[t_k, t'_k]$, they will appear as rectangles in the s-t plane.

This is shown in figure 1(b). Also, viewed as a graph in the s-t plane, the mapping $s(t)$ will be a curve. The **necessary and sufficient condition** for no-collision is that the intersection of the curve, i.e., the graph of the mapping $f(t)$, with these rectangular obstacles, be null. This is analogous to the 2-D static PPP.

Thus, we have reduced the fixed path TPP problem to the following 2-D static problem:

> Given the initial point (s_i, t_i) and the final point (s_f, t_f), find a curve in the s-t plane, avoiding the rectangular obstacles (refer to figure 1(c)).

Note that a more exact computation of the volumes swept by the moving obstacles will give rise to more general shapes than rectangles (in the s-t plane) as shown in figure 1(d).

> Note that the velocity $v(t) = dx/dt$ may be expressed as $v(t) = (dx/ds)(ds/dt)$. For a given curve x, the quantity dx/ds is known. This permits us to restrict ourselves to a simple variable s instead of the vector x, for, determining ds/dt completely determines $v(t)$. Moreover, $|dx/ds| = 1$. Therefore, $|dx/dt| = |ds/dt| \leq V_{max}$. In other words, the same V_{max} constraint holds for $|ds/dt|$. We emphasize that the original space curve x may be any curve in 3-D; we need a parameterization of the curve x by its arc length s. Once the arc length s is known as a function of time, it may be easily mapped back to x as a function of time, since $x(t) = x(s(t))$.

The Static 2-D PPP

There exist several techniques to solve the static 2-D PPP, e.g., (i) the visibility graph [Wang 74, Perez 79], (ii) the generalized cone [Brooks 82], and (iii) the Voronoi diagram [O'Dunlaing 83]. Any of these techniques may be used to solve the static 2-D PPP, which was obtained from the fixed path TPP. We present an approach based on the visibility graph. First, we give a brief overview of the visibility graph approach to solving the static 2-D PPP, and then, extend it to solve the static 2-D PPP (in the s-t plane) corresponding to the fixed path TPP.

The static 2-D PPP:

> Find a path for a point object from its initial position I to the final position F while avoiding the polygonal obstacles.

The **necessary condition** for the minimum length path is that it be composed of straight line segments connecting a subset of the vertices of the polygonal obstacles [Wang 74, Perez 79]. For a proof, refer to [Kant 84]. This 2-D (static) PPP is solved as follows:

Consider the graph, G, with the set of nodes $N = I, F \cup V$, where I corresponds to the initial point; F, to the final point; and V is the set of vertices of the polygonal obstacles. Construct the edge set L of all the edges (n_i, n_j) such that the straight line connecting node n_i to node n_j does not intersect any of the obstacles. This graph $G(N, L)$ is called the **visibility graph** and the shortest collision free (physical) path from the initial point to the final point is given by the minimum cost (graph) path from node I to node F where the cost of an edge is given by the Euclidean metric.

The Fixed Path TPP Algorithm

We have reduced the fixed path TPP problem to the following 2-D static problem:

> Given the initial point (s_i, t_i) and the final point (s_f, t_f), find a curve in the s-t plane, joining these two points, while avoiding the rectangular obstacles (refer to figure 1(c)).

The Constraints The visibility graph in the s-t plane has following constraints:

1. The first constraint is due to the fact that it is impossible to move backwards in time. This translates the graph to a directed graph since a node (s_j, t_j) may be accessible from another node (s_k, t_k) only if $t_k < t_j$.

2. The second constraint comes from the maximum velocity limit. The velocity corresponds to the inverse slope $|ds/dt|$ in the s-t plane. Hence, an edge is valid if and only if the velocity corresponding to the edge is less than or equal to V_{max}.

To accomodate these two constraints, the visibility graph needs to be modified. We state the results here. For the corresponding proofs, refer to [Kant 84].

> It is sufficient to prune all those edges which correspond to velocities less than V_{max}. A minimum path search over the remaining edges will give the optimal path within the V_{max} constraint. If such a path does not exist, it implies that there does not exist a collision-free velocity profile to the goal node.

Two nodes in the pruned visibility graph are connected by an edge if the corresponding velocity (the inverse slope $|ds/dt|$) is less than or equal to V_{max}. A minimum path search over this graph will give rise to the optimal path as shown in figure 2. Note that a straight line segment in the s-t plane will correspond to a constant speed motion. The distance between two points (s_j, t_j) and (s_k, t_k) in the s-t plane will correspond to $\sqrt{((s_k - s_j)^2 + (t_k - t_j)^2)}$. This is equivalent to $\sqrt{(1 + v^2)}(t_k - t_j)$ where $v = (s_k - s_j)/(t_k - t_j)$. The **optimal** path in this visibility graph corresponds to a velocity profile that **minimizes** the above cost function over all possible velocity profiles. This cost function is related to the energy dissipated, but, note that it does not account for switching from one velocity value to another.

The Fixed Path TPP: Time as Cost Function

Another version of the fixed path TPP is one in which the final position, s_f, is specified and the aim is to reach it in minimum time, t_{min}. In the s-t plane, minimum time corresponds to minimizing path lengths with respect to a different cost function, i.e., the cost $[(s_i, t_i), (s_j, t_j)] = (t_j - t_i)$. The goal node may lie anywhere on the vertical line L, (refer to figure 3) passing through the final position s_f. We state the result to obtain the minimum time solution. For a proof, refer to [Kant 84].

For each node in the pruned visibility graph (as obtained in the previous section), construct a probe (i.e., a ray) with slope (V_{max} or $-V_{max}$). If this probe intersects the vertical line L, without intersecting any obstacle first, i.e., if this node can "see" the line L, the point of intersection with the line L, will be a new potential final node (refer to figure 3). The visibility graph is augmented with these new potential final nodes. The node, that may be reached from the initial node, and has minimum time co-ordinate, is the required goal node; the corresponding time is the minimum time solution to reach the final position s_f.

Velocity Profile : Smoothness Requirements Consider figure 2 . The optimal path (from node I to node F) in the s-t plane, corresponds to the velocity function shown in figure 4. The velocity profile is discontinuous. This implies infinite acceleration. This occurs because straight line segments were used to connect the vertices. This results in C^0 continuity. In order that acceleration be finite, we require C^2 continuity. One approach to remedy this is to smoothly interpolate between the vertices (through which the path in the s-t plane passes). It is sufficient to use cubic splines for interpolation to assure C^2 continuity [De Boor 78]. The interpolated function may not intersect the forbidden regions.

Conclusion

We generalized the path planning problem to that of trajectory planning problem in a time-varying environment. An algorithm was developed to solve a special case of the TPP – the **fixed path TPP**, where the path is a point set fixed in space and the robot is an idealized point object. The algorithm determines a velocity profile (along the fixed path) for the robot such that there is no collision with the obstacles. Generalizations of the ideas developed in this paper to deal with the more general TPPs are in preparation.

References

[Brooks 82] Brooks, Rodney, A., Solving the Find Path Problem by Representing Free Space as Generalized Cones, A.I. Memo No. 674, MIT, Artificial Intelligence Lab, May, 1982.

[De Boor 78] Carl De Boor, A Practical Guide to Splines, Springer-Verlag, 1978.

[Kant 84] Kant, Kamal, and Zucker, S., A Computational Framework for Trajectory Planning in Time Varying and Uncertain Environments, Technical Report (in preparation), Dept. of Electrical Engineering, McGill Univ., Montreal.

[O'Dunlaing 83] O'Dunlaing, C., and Yap, Y., K., The Voronoi Method for Motion-Planning: The Case of a Disc, Technical Report No. 53, Department of Computer Science, Courant Institute of Mathematical Sciences, New York, March 1983.

[Perez 79] Lozano-Perez, T., and Wesley, M, "An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles", CACM, Vol 22, pp. 560 - 570

[Reif 79] Reif, John, H., "Complexity of the Mover's Problem and Generalizations", Proceedings of 20th IEEE Symposium on the Foundations of Computer Science, 1979, pp. 421 - 427.

[Schwartz 82] Schwartz, Jacob, T., and Sharir, Micha, On the Piano Movers Problem II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds, Feb 1982, Report No. 41, Computer Science Department, Courant Institute of Mathematical Sciences.

[Wang 74] Wangdahl, Glenn, E., Pollock, Stephen, M., and Woodward, John, B., "Minimum-Trajectory Pipe Routing", Journal of Ship Research, March 1974, Vol 18, No 1, pp. 46 - 49.

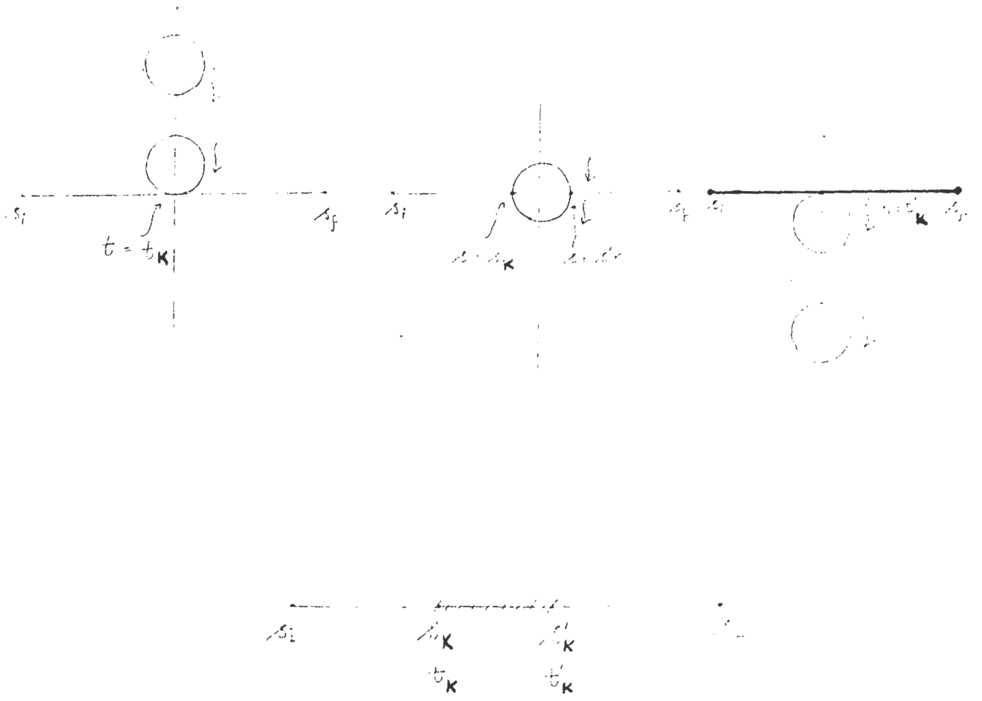


Figure 1(a) Computation of constraints for the velocity function. The obstacle is a circle. It is shown crossing the path segment $[s_i, s_f]$. At $t = t_k$, it touches the path, and at $t = t'_k$, it leaves the path. The maximal subsegment occupied by the circle is $[s_k, s'_k]$. To a first approximation, this subsegment, $[s_k, s'_k]$ is considered occupied or unavailable during the time interval $[t_k, t'_k]$.

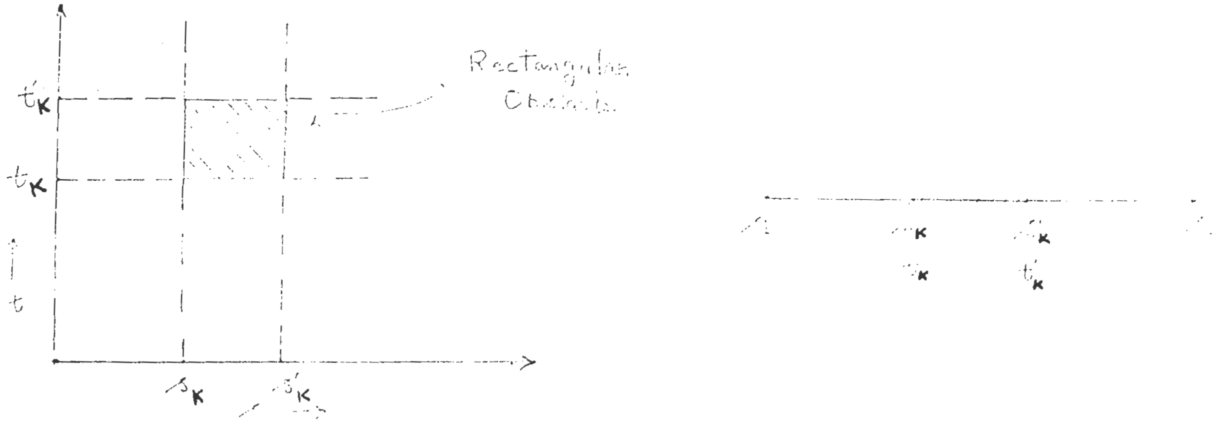


Figure 1(b) The hashed portion, $[s_k, s'_k]$, of the segment $[s_i, s_f]$, is unavailable during the time interval $[t_k, t'_k]$. This is shown in the s-t plane as a rectangle. This rectangle is a forbidden region for the trajectory to pass through.

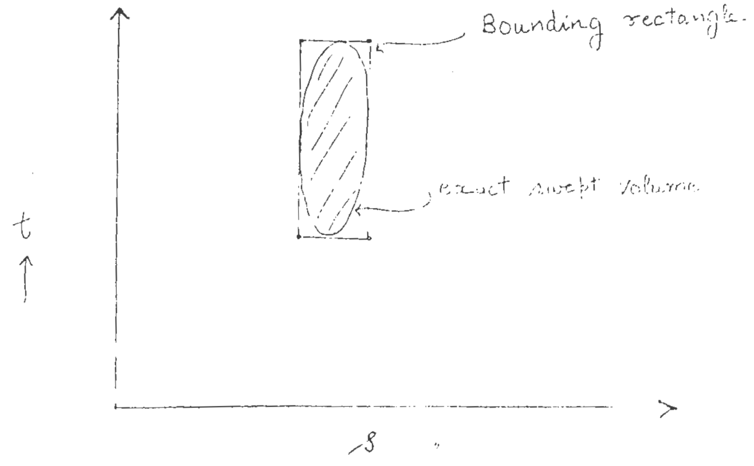


Figure 1(c) A more exact computation of the swept volumes could give rise to more complicated shapes, in the s-t plane, e.g., an ellipse in the case of a circle crossing the path segment. We are taking bounding rectangular approximation to these shapes.

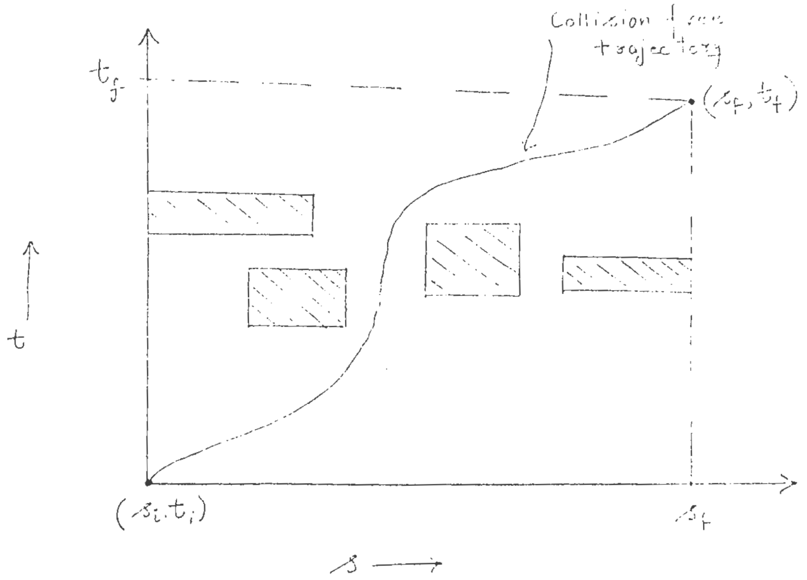


Figure 1(d) Equivalence of the fixed path trajectory planning problem to the (static) 2-D path planning problem in the s-t plane. The horizontal axis is the arc length and the vertical axis is the time. The hashed rectangular regions (computed as in figure 1a and 1b) represent the 2-D obstacles through which the trajectory may not pass.

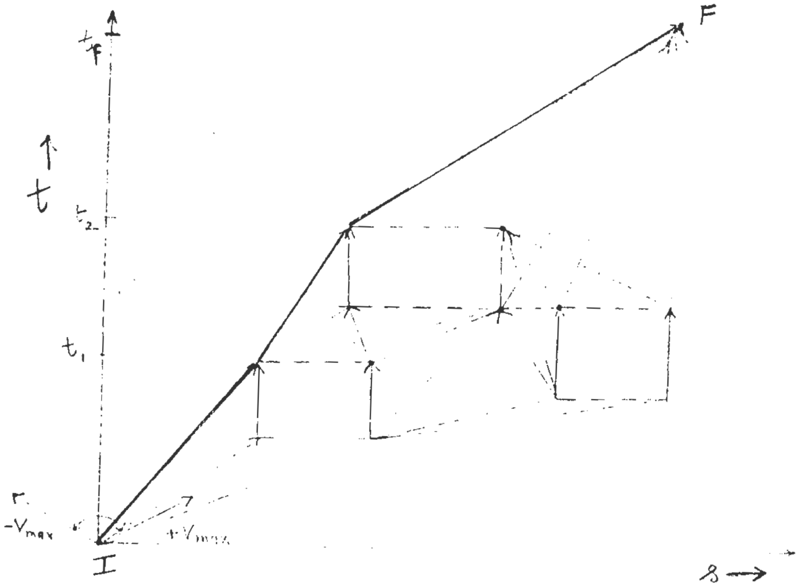


Figure 2 The maximum velocity constraint modifies the visibility graph. In the s-t plane, the inverse slope of an edge corresponds to the velocity of the robot. The edges corresponding to velocity greater than V_{max} are pruned out. These edges are shown in dotted lines. Also the graph is directed (as shown by the arrows) since one can not go back in time. A minimum path search over the pruned graph gives the optimal path in the s-t plane. The minimum path, from I to F, is shown in bold lines.

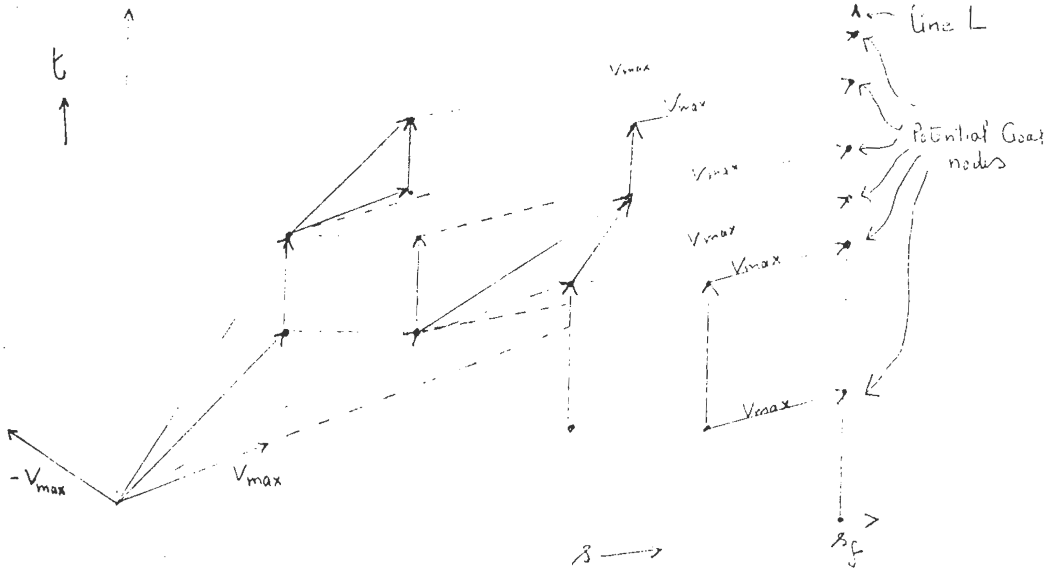


Figure 3 The final position s_f is specified in this case. The time when the robot reaches this position is to be minimized. It is determined by the intersection of a (one of many possible ones) trajectory with the vertical line L that passes through s_f. The $Vmax$ (and $—Vmax$) probes intersecting the line L give rise to potential final nodes. If a probe intersects an obstacle before it intersects the line L, no potential goal node is generated. Such probes are shown in dotted lines. Minimum time corresponds to the node that is reachable and has minimum t coordinate.

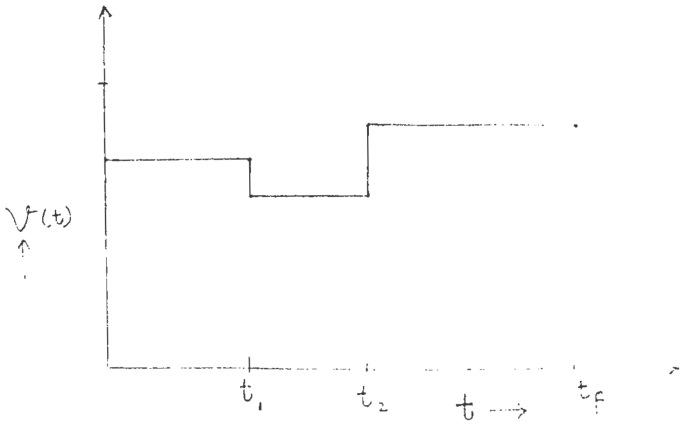


Figure 4 The velocity profile corresponding to the path in the s-t plane in figure 2 . Note that the velocity profile is discontinuous, thereby implying infinite accleration. A smooth cubic spline fit is required. But the spline fit is constrained by the obstacles in the s-t plane, i.e., it may not intersect the obstacles.

Interpreting Range Data For A Mobile Robot

Stan Letovsky
Department of Computer Science
Yale University
New Haven, Connecticut, 06520

Abstract

The Yale Spatial Reasoning Group has been developing perceptual and navigational strategies for use in conjunction with a Heathkit Hero-1 mobile robot. This paper describes a strategy for recovering from sonar range data an approximate description of the environment which generated it.

1 Introduction

Sonar rangefinders are common sensing devices in the current generation of commercial mobile robots. A sonar rangefinder measures the elapsed time between the emission of an ultrasonic pulse and the detection of a reflected echo of that pulse. This elapsed time is proportional to the distance between the sensor and the nearest reflecting object. The technology is simple, cheap, and provides data in a form which is directly usable for some purposes (eg., obstacle avoidance), in contrast to the considerable processing that must be done on, say, visual data, in order to do anything useful. [8] A more advanced, and interesting, use of sonar range data is in the construction of cognitive maps. A cognitive map is a representation of a physical environment, which can be used for navigation and route planning. Representations for cognitive maps have been explored by Davis [2] [3], Kuipers [4] and McDermott [6]. If sonar range data is to be used to construct a cognitive map, the raw range data must be *interpreted* to recover a description of the environment that produced it. The interpretation process is analogous to AI theories of vision [5], [1], in that it involves a computational inversion of the physical processes that generated the "image". This paper describes one method for acquiring and interpreting range data.

2 The Sonar "Image"

The sonar rangefinder's view of the world has several distinctive features. The first is its range limitation. When a sonar rangefinder takes a sample, the sonic pulse leaves the sensor and spreads out in a cone. If the beam is pointed horizontally, as is usual, the sensing is limited by the point where this cone intersects the ground. The distance to this point represents an upper limit on the distances which can be measured with the sensor. In the Hero robot, this distance is about 8 feet.

Secondly, there is an angular resolution issue. When a sample is taken, there is no way to tell what the orientation of the point that produced the echo was, other than that it lay within the sonic cone. For HERO, this cone has a total angle of 30 degrees. Thus, if a sample reads a 6 foot distance, it means that the nearest object in the direction that the sensor is pointing plus-or-minus 15 degrees is 6 feet away.

Finally, since the robot cannot change the vertical orientation of the sensor, the perception problem is considered to be 2-dimensional: the world consists only of open spaces and impassable boundaries, there are no steps or pits.

The way the world appears to such a robot is illustrated in Fig.1 . A sample environment is shown in thin lines, consisting of a hallway with doorways and corners, and a desk. Superimposed on this scene is a possible sonar image of it. The robot, represented by a hexagon, stands in place and rotates a full 360 degrees, taking range samples every 15 degrees. The value measured in each sample is shown as a point whose orientation from the robot is identical to the direction of the sensor when the sample was taken, and whose distance from the robot is equal to the distance measured. Points that were beyond the 8 foot sensing limit are shown as little circles at the limit; the other points are connected by thick line segments.

3 Recovering The World Description

Mathematically, we can describe the distance perceived by the rangefinder with the formula

$$P(\theta) = \underset{\psi=\theta-\phi}{\overset{\theta+\phi}{MIN}} D(\psi)$$

where

$P(\theta)$ ≡ perceived distance as a function of sensor orientation
ϕ ≡ beam semi-angle (15 degrees, for HERO)
$D(\psi)$ ≡ true distance as a function of orientation

This may be inverted as follows:

$$D(\theta) \geq \mathrm{MAX}(\ P(\theta\text{-}\phi)\ ,\ P(\theta+\phi)\ )$$

In words, the larger of the two perceived distances seen at ϕ degrees on either side of θ give a near bound on the true distance to the point at θ. This equation, here referred to as the *back-transform*, represents the mathematical inverse of the simple sonar "image" formation process.

The view of the world which it generates is shown in Fig.2, which was formed from the raw data in Fig.1 using this equation. You will note that the reconstruction is not perfect; information is lost in the original encoding by taking the continuous MIN, and the back-transform cannot recreate this information. The back-transform is better at recovering convexities (eg., the desk corner) than concavities (eg., the doorway or the corner where the desk meets the wall). This knowledge can be put to use. It turns out that the inequality in the back-transform is a strict equality except when the robot is looking into a concavity; thus, perceived concavities indicate departures from completely accurate perception.

One feature which distinguishes the back-transform from the sonar image can be seen from examining the lines that connect the sample points. In the sonar image, these lines describe a region that is completely free of objects, so that if the robot stays within this region it will never bump into anything. In the back-transform this useful property is lost. The reason is that the back-transform is finding the true distance at particular points, and has nothing to say about the points in between them. So in those intervals where concentrating exclusively on the back-transform might lead the robot to bump into things, the sonar image still contains useful information.

4 Implementation Notes

The images in this paper, and the theory behind them, were developed using a simulator written in T on an Apollo workstation. When our HERO was assembled, we discovered that its rangefinder performance fell considerably short of the specs. Its data is of poor quality, often spurious, and extremely susceptible to environmental ultrasonic noise. Also, it has been suggested to me [9] that its results are a function of angle of incidence and surface texture. Due to these problems, this strategy was never successfully implemented on Hero. However, more recent work by Miller [7] shows promise for coping with these difficulties.

5 Conclusion

A sensing strategy and data-analysis technique has been described which permits approximate reconstruction of the true description of a 2-dimensional environment from sonar-range samples of that environment. Like perceptual methods developed for other senses, the technique recovers the original environment by inverting the physics of the sensing process.

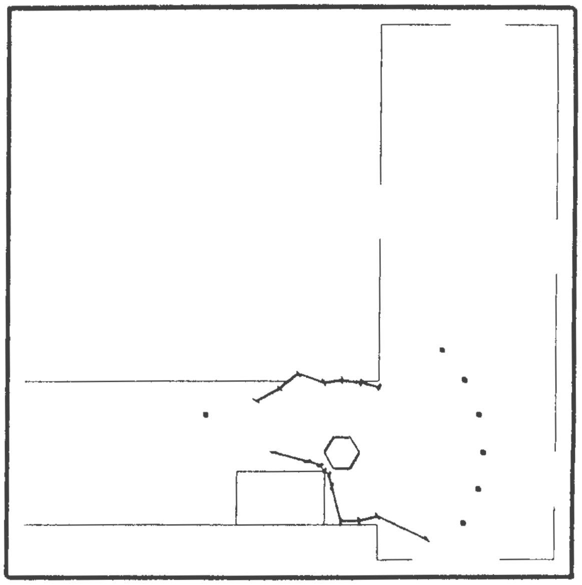


Figure 1: A hallway with desk, doorways, robot, and sonar "image".

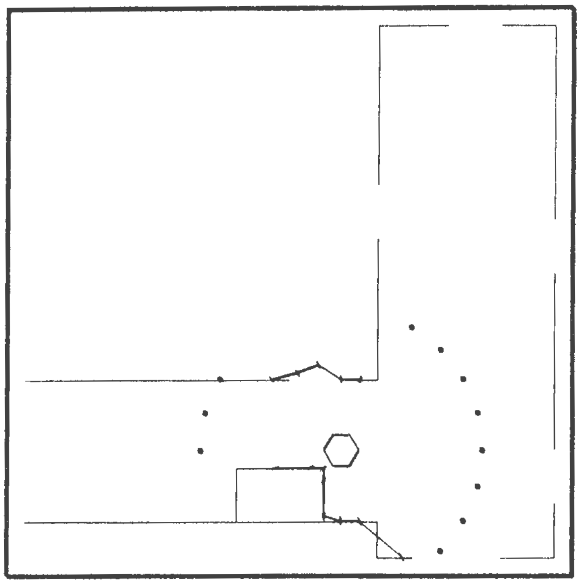


Figure 2: Back-transform of Fig.1.

6 Acknowledgements

The author thanks Drew McDermott and David Miller for their contributions to this work.

References

[1] Barrow, H. and Tenenbaum, J.M.
Computational Vision.
Proceedings of the IEEE 69(5):572-595, 1981.

[2] Davis, Ernest.
Organizing Spatial Knowledge.
Technical Report 193, Yale University Computer Science Department, 1981.

[3] Davis, Ernest.
Reasoning and Acquiring Geographic Knowledge.
Technical Report 292, Yale University Computer Science Department, 1984.

[4] Kuipers, Benjamin.
Modeling spatial knowledge.
Cognitive Science 2(2), 1978.

[5] Marr, David.
Vision.
W.H.Freeman, 1982.

[6] McDermott, Drew V.
Spatial inferences with ground, metric formulas on simple objects.
Technical Report 173, Yale University Computer Science Department, 1980.

[7] Miller, David.
Mobile Robot Positioning From Sparse Sonar Data.
In *Proceedings, 1984*. AAAI, 1984.
Submitted.

[8] Moravec, Hans P.
Rover Visual Obstacle Avoidance.
In *Proceedings, 1981*, pages 785-790. IJCAI, 1981.

[9] Schultz, Hayden.
Personal communication.
Department of Electrical Engineering, Purdue University.

THE USE OF CAUSAL EXPLANATIONS IN LEARNING[a]

David J. Atkinson and Steven Salzberg[b]
Yale University
Artificial Intelligence Project
Department of Computer Science
New Haven, Connecticut 06520

Abstract – Models of learning in complex domains are faced with the difficult task of sorting through the masses of data which humans must examine to understand those domains. The use of a *causal model* is necessary to constrain the search process through these domains, where a causal model includes knowledge of the relationships between the fundamental events and objects in a given domain. Causal knowledge is particularly useful in AI systems where the task is to predict future events in some complex domain. When a prediction fails, causal knowledge is used to examine the relevant data and create an explanation of the failure. This explanation then drives memory reorganization processes in the system. Two case studies provide examples of the use of causal knowledge in processing: FORECASTER, a program which predicts the weather, and HANDICAPPER, a program which predicts horse races.

Why Explanations are Necessary

In any real world domain, human experts know an enormous number of facts. AI researchers have often in the past avoided such domains precisely because the number of facts is unmanageably large. Current programs, however, are beginning to tackle these types of domains ([17, [9], [8], [13]), and new techniques are being developed to handle the new types of problems that are arising.

Whether a program performs a task, predicts an event, or diagnoses a problem, it still will fail at times. Failures are not a bane for AI systems, however, because they provide an opportunity to learn something new, in order to modify the behavior of the program when a similar situation occurs in the future [14]. The first question which a person or program must ask after a failure is, naturally, why did the failure occur? The answer must be discovered by examining what factors influenced the decision in the first place, and changing some of those factors or their relationships to one another.

Take for example a model that predicts earthquakes. Two types of failures occur here, which are typical of AI prediction models: either an event happens which was not predicted, or an event is predicted and it does not happen. For earthquakes the consequences are much more serious in the former instance, but in general we want our models to respond to both types of failures. The problem that human experts have been struggling to solve is what events precede an earthquake. Unfortunately, there are thousands of environmental events that might potentially cause an earthquake: other earthquakes, volcanic eruptions, movements of the tectonic plates, changes in sea level, underground atomic tests, sunspots, the alignment of the planets, the position of the moon, and so course. A recent *New York Times* article described a theory held by some geophysicists that earthquakes at widely separated places on the planet might be connected to each other. In order to make such a claim, the events preceding each of the separate quakes must be traced back to some common event. What the geophysicist wants, at least after an earthquake occurs, is a model of geophysics that provides an explanation of the earthquake. An explanation consists of *a sequence of events whose causal relationships are clearly understood.* At the very least, explanation consists of identifying the plausible proximal causes of an event. Of course, it is impossible to prove that any event really causes any other, but certain relationships must simply be axiomatic in any domain. Continental drift theory, for example, might be among the axiomatic rules in a model of earthquakes. While people typically prefer explanations of this form; i.e., based on unitary causation [6] [12], they also acknowledge (but less frequently act upon) explanations based on multiple causes [18].

Explanations are necessary because without them we have no organized way of changing our models after failures. Learning in any complex domain is simply impossible if we have no knowledge to direct the search through the space of features that may have caused a failure [11]. In our earthquake example, suppose we had twenty (not necessarily unrelated) geophysical events preceding a major quake. Without any knowledge to direct it, a model cannot distinguish between the hypothesis that any single event or any *combination* of events caused the quake. The number of combinations of twenty events is unmanageably large, so obviously we need to use whatever causal knowledge is available to build explanations. Although one could use a syntactic rule like "generate the shortest hypothesis" to avoid the *hypothesis explosion* problem, this rule fails for obvious reasons in any situation where a complex explanation is the best one. Suppose the correct reason for a given earthquake were a combination of five different factors: a program which generated the shortest explanation would

(a) This work was supported in part by the U.S. Air Force Office of Scientific Research under contract F49620-82-K-0010.

(b) The order of names of the authors is not significant.

generate (at least, if no other factors were known) thirty incorrect explanations before it reached the combination of all five factors. It is unlikely that such a program would ever have thirty similar earthquakes to refine its model, so we clearly want to generate the correct explanation sooner.

One more example should suffice to prove the necessity of explanations. Suppose the domain were the stock market, and the task were to predict the behavior of a certain computer company, call it CCC. If the stock goes down when it was expected to go up, one has to look at why it was expected to rise. There may have been numerous factors: perhaps the personnel are known to be good, and maybe a new product has gotten favorable reviews. The general climate for computers might be good as well, but the new product in this case was the principal reason for expecting the stock to rise. Suppose, when the explanation process looked back over its decision, all the factors used to make the prediction seemed sound. One of the factors about the marketplace, however, was that IBM was releasing a very similar product. Here is where causal knowledge can play a role. Why, one should ask, might a similar product on the market cause CCC's product to sell poorly? Causal knowledge about the marketplace includes the rule:

```
If two products are equally good,
   and one product sells better than the other,
   then the company producing the better seller
   has better marketing strategies.
```

The hypothesis which must be produced, then, is that IBM has better marketing strategies than CCC. If this is the case, then the new product will not sell well, and therefore the stock of CCC will not rise. By using the available causal knowledge, the model can focus on the source of the failure, and learn something new (about IBM) in the process.

Having learned that IBM has better marketing abilities than CCC, the model should be reminded of this example by future, similar events. When another company comes along with a product, and IBM is also marketing that product, the model will know that IBM's product has a greater likelihood of selling well because of IBM's superior marketing ability. The reminding will occur because the entire episode above will be stored under indices that are used to process the episode [14]. The indices can be generalized so that the new knowledge can be used in more general situations: anytime two companies with similar products are competing, the model will look for the company with better marketing strategy, and predict that company's product will sell better.

Without an explanation process, it would be impossible to learn the fact learned in the above example. All of the factors contributing to the initial prediction would be cast in doubt, although in fact none of them were incorrect. The explanation involved hypothesizing a *new* factor, marketing strategy, and attributing the failure to it. A more inductive process without causal knowledge would be forced to choose some or all of the factors and blame them for the failure, but it would simply be wrong to say, for example, that a good product should not lead to a rise in a company's stock. The explanation process singles out one line of reasoning which leads to plausible changes in the model of the domain. Because the process uses causal knowledge, it avoids wrongly hypothesizing that events which were believed to be connected are not. Furthermore, the hypotheses generated by explanations are not necessarily correct, but can be tested empirically, and can be revised in the light of future examples. Since pointers can be kept to previous episodes, the revision process can be designed to insure that a consistent model is maintained at all times.

How Explanation is Done

There are at least three primary purposes to explanation for which algorithms must be developed. First, as discussed above, explanation is a procedure which utilizes causal knowledge in order to limit the number of hypothetical causal factors relevant to a failure. Without the judicious application of causal knowledge about the domain in question, the number of speculations about the causes of a failure can be phenomenally large with the complexity of the domain -- and the real world is certainly complex. To test and verify this number of hypotheses in real world experience (i.e., not through experiences thoughtfully provided by a teacher) is computationally intractable given our present understanding.

Explanations should improve

Secondly, explanation procedures should allow prediction failures in certain circumstances to signal an orderly change in the causal model of the domain. Without this facility, explanations could never improve beyond those possible with any *a priori* causal model of the domain in question. It is unlikely, however, that a complete causal model of all the domains of experience is innate. The explanations provided by humans clearly improve with experience in a given domain, and it is our contention that this is the result of failure-driven modifications to memory. If we want our AI programs to improve in their understanding of a domain -- especially if we want their understanding to improve beyond our own -- the re-organization and growth of causal models needs to be a focus of research.

Avoiding future failures

Finally, the most important function of explanation is to provide a mechanism whereby future expectation failures can be avoided. Having made an adequate explanation for a failure, and perhaps having reorganized some causal knowledge, the causal elements leading to the correct expectations must be identified *in advance* of any prediction failures. It is not enough to improve understanding of the reasons for an erroneous prediction if it cannot be avoided in the future. Hence we need to make failure- and explanation-driven modifications to the knowledge structures which represent and process events.

Deduction of missing information

In the CCC example in the previous section, the causal relation that ***superior marketing strategies are likely to result in higher product sales against a similar product*** is known, but the data about the marketing strategies of the companies in question is unknown. This defines one type of condition for explanation, i.e.; where the relevant causal relation is known, but the data which would cause the relation to be applied in the current instance is unknown. In the absence of competing explanations, the deduction that IBM has superior marketing strategies provides a forthright explanation.

Furthermore, the prediction which failed could have taken place in two different contexts regarding the information about marketing strategy. First, the processing structure responsible for the prediction could represent the fact that knowledge about marketing strategy is important in making a prediction about product sales, but the real data about IBM and CCC are unknown. The utility of making predictions from partial data in the absence of complete data is one of the motivations for frame based representations and content-addressable memory (e.g., [2]). The hypothesis is that what is known is ***sufficient*** to make the inference. In this case, however, the explanation process has shown that without the data about marketing strategy, an inference about future product sales is unreliable. Therefore, the current processing structure should itself be re-indexed such that data about marketing strategy is required before the structure is used for prediction. This effectively prevents a prediction failure from recurring. Finally, a representation of the current situation is constructed, and indexed from the current structure by a specification of the type of failure.

Adding to incomplete causal models

Suppose that at some later time, the episodic processing structure that we have been discussing is retrieved. This time, CCC is introducing a microcomputer which is again similar to another company's product, and it is known that CCC's marketing strategies are slightly superior to its competitor's (call them TEX). The prediction is made that CCC's product will sell more, but once again the prediction fails. A search of causal knowledge reveals that there are no additional rules to apply which can help identify elements in the current situation which may have caused the failure. However, since the current failure is similar to the previous one, a *reminding* occurs of the previous failure [14].

The situation we are describing is one where a relevant causal relation has not been retrieved from the causal model of the domain, and it is unknown whether or not the causal factors relevant to the failure have been observed. The task of explanation processes in this case is to try to identify ***previously unknown*** potential causal factors in the current failure. It is in this situation that one utility of the reminding process is made apparent. An explanation procedure which may operate here compares the two episodic representations in order to find similarities and differences. Let us suppose that both IBM and TEX are located in the southwest, and CCC is located in the northeastern part of the country. A hypothetical causal explanation is that location in the southwest is important (in the context of the other variables) for strong product sales.

This causal *explanation* becomes a hypothesis about a new causal *relation* to add to the existing model of the domain. This process of case-based induction is related to the logical methods of discovering causal regularities called "eliminative induction", first proposed by J.S. Mill [10] (and more recently described in [7]). Functionally, the hypothesis may serve in memory as part of an index to the episodic structure which represents a generalization of the two failure episodes. The matching of this index in future episodes provides a test of the hypothesis. Successful prediction of product sales based on the information supplied in this new processing structure signals that the explanation linking southwest location to product sales may be added to the causal model of the domain. This example should make clear the strong interdependence of the causal model of the domain and the structures in episodic memory which organize and predict events.

To clarify things further, let us turn now to case studies of two programs that produce explanations as a result of failures, and use these explanations to improve future performance.

Example: FORECASTER

FORECASTER is a program which operates in the domain of weather prediction. The goals of the research of which the program is a part are to explore the initial development of episodic and causal knowledge using explanation processes. The FORECASTER program processes synoptic weather reports obtained from a National Weather Service station and attempts to predict the future weather reports for that station. The program builds simple memory structures to represent short sequences of weather reports that it has seen, where a typical report consists of about seventeen different kinds of observations. These episodic structures are stored in memory using indices derived from the particular weather observations represented in the first report in the structure (e.g., Temperature 15°, Visibility poor). In particular, an index specifies the conditions which the program believes are causally sufficient for expecting the observations in the subsequent weather reports in one of these structures.

When the program receives a new weather report, the indices to structures are checked. When an index has a match among the current observations, it causes the corresponding episodic structure to be retrieved and then used to make predictions about the next weather report from the given station. This process of matching new observations against indices is a process of judging when a new weather report is similar to one which has been experienced before. If the same ***causally relevant*** observations are present in the new weather report that were present in a previously experienced one, it is a natural inference that subsequent observations will also be the same. FORECASTER does not use causal knowledge to make predictions of individual observations, but rather of weather reports as a whole.

One example of FORECASTER's explanation abilities occurred when it was processing the data from the Albany, New York weather station in November 1983. It illustrates the most simple way in which the program hypothesizes and uses causal regularities in the weather reports it processes. In this case, an episodic memory structure was retrieved on the basis of several different observations, including the height of the cloud base, type of low clouds, overcast sky, and visibility. However, the retrieved structure did not reflect all of the current observations. The temperature and dewpoint values were among those observations which were not matched; the temperature was slightly lower and the dewpoint was quite a bit lower than the corresponding values represented in the first report in the structure. However, because they were not part of the active index, they were not considered ***necessary*** to predict the subsequent observations in the structure. So, the program went ahead and predicted the next report from that station. The predictions included that the base of the clouds would be below 100 feet, and both the dew point and temperature would be around 2 degrees Celsius.

When the next weather report was obtained, the program had several failed predictions, including that the cloud base was about a hundred feet higher than expected. Like the CCC example above, an explanation strategy used by the program is to examine the supposed causal antecedents for the expectation. The most basic explanation for this failure is that the observed antecedents, i.e., those observations which were part of the active index, may not be causally sufficient for the prediction of the height of the cloud base. Some other observation, part of the first weather report in the structure, must necessarily be observed before the prediction can be made. Through the use of several heuristics to attribute causality (similar to Mill's methods of eliminative induction, cited above) and repeated observations of co-occurance [1], the program had by this date formed the hypothesis that the value of the dew point and the predicted height of the cloud base could be causally related. It had not yet learned that the temperature of the air is also a factor in this relationship (dew point and temperature together help to determine the saturation of the air, and thus the altitude where the air is cold enough such that condensation will occur). Among all the differences between the current weather report and the observations represented in the retrieved structure, this observed regularity allowed the program to focus on the difference in the dewpoint observation as a probable cause of the failure.

To prevent a failure to accurately predict the height of the cloud base for similar reasons (i.e., an errant dewpoint observation) from occurring in the future when using the same structure, the current index to the structure was specialized. The program did this by adding the *expected* dewpoint observation to the current index, thus making it a necessary condition for retrieval of the structure. In addition, to help future explanations a new episodic structure representing the actual weather data in this situation was indexed "underneath" the current structure by a specification of the failure to predict the height of the cloud base. If a similar failure were to occur in the future when using the current memory structure, then the program would be reminded [14] of this instance of failure.

Sometime in the future, let us suppose that the current weather report matches the dewpoint and other observations as specified in the index that FORECASTER constructed. The same episodic structure would be retrieved once more. Again, the temperature fails to correspond to the temperature represented in the structure but this is an insignificant difference from the point of view of the program. In this case, the prediction about the future height of the cloud base (among 15 other expectations) would again be made.

If the prediction about the height of the cloud base were again to fail, then a reminding would occur of the previous failure. The program would once again examine the context of the failure to check whether the known causal antecedent observations were present (i.e., the dewpoint, overcast sky, and type of low clouds). In this case, the program discovers that they all were observed. In comparing the case of the previous failure and the current case of failure, the program would notice that in both instances the temperature observations were anomalous. However, many such similarities could exist, any one of which could be a candidate cause of the failure.

This set can be substantially reduced by eliminating the observations which are known from experience to be unnecessary for the predicted observation, and by using general domain knowledge to filter additional unlikely causal factors. This comparison-based explanation process reveals that the temperature value is likely to be an important causal antecedent of the height of the cloud base in addition to the dewpoint value. Re-indexing of the current structure would occur as before.

Example: HANDICAPPER

HANDICAPPER is a program which uses causal knowledge to constrain its hypothesis generation process in the domain of horse racing [13]. It has the ability to recognize 35 different features for each horse in a race, and its goal is to pick the correct winners of races. When it picks incorrectly it generates an explanation of its failure. This explanation serves as a hypothesis which is then used in future predictions if similar circumstances recur.

The example to be discussed here occurred in a race in September 1982. The two horses favored by the program both had a feature called "early speed", meaning they usually jumped out to a lead early in a race. The race in question was a short race, and the program knows that early speed is a good feature in a short race, because often the other horses in the race do not have time to make up the distance they've lost at the beginning. In this race, however, another horse, which did not have early speed, beat both of the top horses predicted by HANDICAPPER. The task of the program was then to determine how, if possible, this latest horse could have defeated two seemingly better horses. Knowing that early speed is a good feature in short races, it would seem wrong to hypothesize the opposite, unless there was some causal knowledge that would tell the program otherwise.

There were, in fact, three horses in this race with early speed, none of which won. HANDICAPPER knew that its original prediction was based in part on the fact that the horses it liked had early speed, so it proceeded to simulate the effect on each horse in the race of having three early speedsters. It knew that when one horse has early speed, other horses tend to run a little harder (i.e., use a little more energy) early in the race in order not to fall too far behind. It also knew that horses with early speed run a different style of race from average thoroughbreds: they tend to use more of their energy in the beginning of the race, and then just hold onto their lead if possible through the middle stages and the finish. Combining these two pieces of knowledge, the program discovered that when more than several horses with early speed are present, there is a chance that they will all use too much energy at the beginning and die before the finish. If this happens, it will allow a horse without early speed to defeat them all. For the race in question, this explanation seems adequate, so it is used as the explanation for the failure. A structure for races containing multiple early speedsters is then built, and this race is indexed under it. In future races with more than one early speed horse, this structure will be re-activated, and the advice it contains will prevent the program from using early speed as a predictor of a win.

What the program has learned here is that two horses with early speed in the same race may tire each other out, causing both to lose. It did not know this in the beginning, and it was not designed to discover this single fact. In the beginning, it knew

1. what style of race a horse with early speed runs

2. how a horse with early speed affects other horses in the race

but it had never considered how a horse with early speed might affect another horse with early speed, because such a situation had not occurred previously in its experience. By storing away this experience so that it will be recalled in similar future situations, the program avoids repeating its failure.

Conclusion

We have seen through this discussion and the case studies that followed the value of causal explanations in models of learning. The knowledge of how events in any domain are causally related allows our models to focus more precisely on the sources of error when their predictions fail. For AI prediction models, especially those in complex domains, it is essential that some means be employed to search through the enormous number of potentially relevant causes of any failure, to avoid the hypothesis explosion problem. We have seen how a causal model helps guide this search from the reasons for the initial prediction to the source of the error, and how new causal knowledge can be acquired. The explanations that are built as a result of failures improve the accuracy of the programs' domain models, and subsequently improve their abilities to make accurate predictions.

References

[1] Becker, Joseph D., A Model for the Encoding of Experiential Information, In Schank, R. and Colby, K. (eds.), *Computer Models of Thought and Language*, W. H. Freeman and Company, 1973.

[2] Bobrow, D, and Norman, D., Some Principles of Memory Schemata, In Bobrow, D. and Collins, A. (eds.), *Representation and Understanding*, Academic Press, Inc., (1975).

[3] Burstein, M., "A Model of Learning by Incremental Analogical Reasoning and Debugging", *Proceedings of AAAI-83, Washington, D.C.*, (August, 1983), pp. 45-48.

[4] De Kleer, J. & Brown, J.S., "The Origin, Form, and Logic of Qualitative Physical Laws", *Proceedings of the Eighth IJCAI, Karlsruhe, West Germany*, (August, 1983), pp. 1158-1169.

[5] Gentner, D., "Structure Mapping: A Theoretical Framework for Analogy and Similarity", *Proceedings of the Fourth Annual Conference of the Cognitive Science Society, Ann Arbor, Michigan*, (August 1982), pp. 13-15.

[6] Kanouse, D.E., Language, labeling, and attribution, In E. Jones, et. al. (Eds.), *Attribution: Perceiving the Causes of Behavior*, General Learning Press, (1972).

[7] Mackie, J.L., *The Cement of the Universe: A study of causation*, The Clarendon Press, (1974).

[8] Michalski, R., A Theory and Methodology of Inductive Learning, In Michalski, R., Carbonell, J., and Mitchell, T. (eds.), *Machine Learning*, Tioga Publishing Co., (1983).

[9] Michalski, R., and Baskin, A., "Integrating Multiple Knowledge Representations and Learning Capabilities in an Expert System: The ADVISE System", *Proceedings of the Eighth IJCAI, Karlsruhe, West Germany*, (August, 1983), pp. 256-258.

[10] Mill, J.S., *A System of Logic*, 8th edition (reprinted), (1941).

[11] Mitchell, T., "Learning and Problem Solving. Computers and Thought Lecture", *Proceedings of the Eighth IJCAI, Karlsruhe, West Germany*, (August, 1983), pp. 1139-1151.

[12] Nisbett, R.E. & Ross, L., *Human Inference: strategies and shortcomings of social judgment*, Prentice-Hall, Inc., (1980).

[13] Salzberg, S., "Generating Hypotheses to Explain Prediction Failures", *Proceedings of AAAI-83, Washington, D.C.*, (August, 1983), pp. 352-355.

[14] Schank, R., *Dynamic Memory: A theory of reminding and learning in computers and people*, Cambridge University Press, (1982).

[15] Schank, R. & Collins, G., "Looking at Learning", *Proceedings of ECAI-82, Paris, France*, (1982), pp. 11-18.

[16] Soloway, E. and Riseman, E., "Levels of Pattern Description in Learning", *Proceedings of IJCAI-77, Cambridge, Massachusetts*, (1977), pp. 801-811.

[17] Utgoff, P. & Mitchell, T., "Acquisition of Appropriate Bias for Inductive Concept Learning", *Proceedings of AAAI-82, Pittsburgh, PA*, (August, 1982), pp. 414-417.

[18] Wilson, T.D., & Nisbett, R.E., "The accuracy of verbal reports about the effects of stimuli on evaluations and behavior", *Social Psychology*, (1977), *41*, 118-131.

Experiments in the Automatic Discovery of Declarative and Procedural Data Structure Concepts

Mostafa Aref
Department of Electrical Engineering
University of Toledo
Toledo, Ohio 43606

Gordon McCalla
Department of Computational Science
University of Saskatchewan
Saskatoon, Saskatchewan S7N 0W0

ABSTRACT

A discovery system has been implemented which learns both declarative and procedural concepts in the domain of data structures. The system starts with a small initial knowledge base containing 1 structure ("list"), 1 operation ("search"), 7 relations (such as "lessp", "atom", "memq", etc.), and 26 heuristics. From this knowledge base, the heuristics are able to generate many new concepts, including various kinds of ordered lists, trees, and forests; several sorts of search, including tree search and binary search; and many new relations such as "greaterp", "less-than-all", "not-memq", etc. Heuristics cannot as yet be learned, but the system has been designed with learning new heuristics in mind. It may even eventually be possible to apply the discovery program to itself, and thus automatically discover new things about discovery.

1. Introduction

Learning has finally begun to take its place as one of the cornerstones of artificial intelligence. Evidence for this includes the recent publication of a collection of papers on learning (Michalski, Carbonell, and Mitchell [1983]), the many papers presented at the 1983 machine learning workshop (IMLW [1983]), and the number of sessions devoted to the topic of learning at current AI conferences (e.g. IJCAI [1983]). In these various collections a number of different categories of learning are identified including learning by analogy, learning from examples, concept learning, learning from observation, and discovery. Discovery is the topic of this paper.

Specifically we are interested in automating the discovery of computer science concepts, in particular the discovery of concepts having to do with data structures and their manipulation. Discovering such concepts requires the ability to learn procedural information (relations, operations) in addition to declarative information (structures). Some current work is concerned with the discovery of computer science concepts (e.g. Lenat's [1982b] general discovery system EURISKO which has learned about aspects of LISP programming among other things). However, discovery in the domain of data structures is still an area ripe for further concentrated exploration. Being able to learn new kinds of structures (e.g. lists, trees, etc.) and new kinds of procedures (especially search techniques) is an essential prerequisite to a discovery program being able to discover new things about discovery programs. Moreover, data structures is a relatively shallow area (as compared to mathematics, say, which Lenat's [1977] AM explored), so it is even conceivable that eventually new, useful computer science concepts could be discovered.

Various approaches to discovery have been proposed over the years, notably the data manipulation approach to scientific discovery of the BACON systems (Langley [1980]) and the heuristic search paradigm developed in systems like AM, EURISKO, and LEX (Mitchell [1983]). For our particular application, the heuristic search approach is more appropriate since the concepts being manipulated are more procedural than the data driven approaches can readily handle. In the heuristic search approach an initial knowledge base of concepts (representing various "primitive" data structures and procedures) are manipulated by a number of heuristics that slowly extend this knowledge base until new and interesting concepts emerge.

In our system there are four different types of concepts: operations, relations, structures, and heuristics. The system is able to discover new operations, new relations, and new structures. The system currently does not learn new heuristics, although it has been designed to make such an extension relatively easy. Many systems (such as EURISKO, LEX, and SAGE (Langley [1983])) have explored the discovery of heuristics for various domains. Lenat [1982a] even proposes that the time is right for a theory of heuristics: "heuretics". The insights provided by this body of research will hopefully prove to be useful when extending the current system to be able to discover heuristics for the data structures domain. Such an extension will be crucial if discovery about discovery is to be achieved.

2. Details of the System

In this section some of the details of the discovery system will be discussed. Further elaboration can be found in Aref [1983]. Much as in AM, the four different types of concepts (operations, relations, structures, and heuristics) are represented in frames, a different type of frame for each category of concept.

Operation frames represent various processes that can be undertaken. Here is a simplified version of a typical operation frame, "search":

```
Name: search
Isa: operation
Definition:
    Semantic: looks for an element in a structure
    Parameters: OBJ, LST
    Code:
        Precondition: OBJ is an atom; LST is not an atom
        Succeed: the first element of LST is an atom; and
                 the first element of LST equals OBJ
        Fail: there are no elements in LST
        Recursive: search for OBJ in first element of LST;
                   search for OBJ in rest of LST
        Worth: 200
```

(miscellaneous other slots are not shown; code is pseudo-code)

The key slots (and subslots) are "Isa" and "Definition". The "Isa" slot merely places the frame appropriately in an "Isa" hierarchy. In the early stages of discovery only the top levels of this hierarchy exist; the discovery process slowly deepens the hierarchy as these initial abstract frames are specialized. The "Definition" slot defines the meaning of the concept represented by the frame. In an operation frame this slot has 3 subslots: "Semantic", a comment (in English) describing the action of the operation; "Parameters", a list of the parameters taken by the operation; and "Code", which actually contains the code that carries out the operation (for readability, the code shown here is pseudo-code; it is actually LISP code in the implemented system).

"Code" breaks down into 4 important subparts. "Precondition" defines conditions which must be met by any arguments before the operation can be executed. "Succeed" and "Fail" are base conditions for the recursion defining what it means to terminate successfully or not in the execution of the operation. "Recursive" is the recursive step that is undertaken if the termination conditions are not met. Breaking down the action of an operation into parts allows the discovery process to manipulate each part separately and hence "understand" some of the subtleties underlying the structure of a program.

The other 3 types of frames are similar to operation frames. Relation frames represent predicates and are much like operation frames except that the "Code" subslot is simpler (no recursive definition is necessary). There is also an "Examples" slot that contains data structures for which the relation is true (e.g. pairs of numbers where the first is "lessp" the second). Here is a version of one such relation frame, "lessp":

```
Name: lessp
Isa:  one-to-one
Definition:
    Semantic: return true if the first number is less
                than the second number
    Parameter: NUM1, NUM2
    Code:
        Precondition: NUM1 and NUM2 are numbers
        Relation:  NUM1 is less than NUM2
    Examples:  (2 5), (22 202), (73 83)
    Worth: 100
```

Structure frames represent various data structures. In a structure frame the "Code" slot is a recursive definition of the structure, broken down into slightly different subparts than an operation frame. In addition there is an "Examples" slot that is used to contain examples of the structure. Usually, these can be generated by the system from the definition; if for some discovered structure no examples can be generated, the structure can be rejected as not meaningful. The system's "list" frame is shown below:

```
Name: list
Isa:  structure
Definition:
    Semantic: a collection of non-repeated elements
    Parameter: L
    Code:
        Precondition: L is not an atom
        Termination: the first element is not an atom; and
                       there are no more elements in L
        First-step: the first element is an atom; and
                      the first element is not in the rest
                      of L
        Recursive: the rest of L is a list
   Examples: (kxfup c s m 189 219 788 771), (777)
   Worth: 200
```

Heuristic frames are the driving force behind the discovery process in that they actually create the newly discovered frames. The "Code" slot of a heuristic frame works with one (or more) frame(s) of other types. The action of the code is to generate a revised version of the frame(s) provided to it, perhaps indirectly through the agenda --- see below. Here is a simplified version of the "add-relations" heuristic frame:

```
Name: add-relation
Isa:  h-structure
Definition:
    Semantic: if the current task is to add a relation,
                and the structure hasn't a specific type
                of relation, then add tasks to the agenda
                to try to add each type of relation to the
                structure
    Code:
        If-part: if the task is to add a relation to a
                   structure S
        Then-part: add to the agenda tasks to add relations
                     to S which are not in S
Worth: 50
```

Here is an example of how the heuristic frames act in concert to produce the learning behaviour of the system. The "change-relation" heuristic" might start off by suggesting that the preconditions for each element of a "list" structure be "element is a number" rather than "element is an atom". This would result in a new concept, "list-of-numbers", being generated. If examples could be generated of "list-of-numbers" then the new frame could be added to the Isa hierarchy as a sub-concept of "list". Next, the heuristic "add-relation" might insist that the relation "lessp" must hold between each pair of elements in "list-of-numbers". Adding this restrictive relation would give an "ascending-ordered-list-of-numbers", assuming once again that examples could be generated. Another heuristic, "change-domain" might then suggest changing the preconditions on the parameters for "search" so that LST must be an "ascending-ordered-list-of-numbers". If the heuristic also made minor changes (mostly trivial lexical changes) to the rest of the "Code", the frame could be successfully executed and a new concept "search-ascending-ordered-list-of-numbers" could be added as a sub-concept of "search".

This is fairly typical of the kind of learning which goes on in the discovery system. Small modifications of various kinds to the "Code" parts of existing frames yield new frames. These new frames are further modified (sometimes involving references to other new frames), and so on, until startingly different concepts can eventually evolve. Of course, there are many irrelevant steps and blind alleys along the road. There is also the chance that some heuristic can postulate many different possible modifications. Thus, there must be some way to handle multiple goals and to choose which goals are most promising.

This is accomplished using an agenda mechanism similar to that employed in AM. This agenda is a list of "tasks", each of the form

```
(worth concept-to-be-modified suggested-heuristic
          <further-specification>)
```

arranged in descending order of the worth of the tasks. The concept-to-be-modified is the name of a frame that may be modifiable to get a new concept; the suggested-heuristic should be able to make the appropriate modifications. An optional further-specification can impose a limitation on the action of the suggested heuristic. An example of this might be the "add-relation" heuristic which, in addition to trying to add "lessp" to "list-of-numbers" might also consider adding "greaterp" or "equal". To do this, the tasks

```
    (worth1 list-of-numbers add-relation lessp)
    (worth2 list-of-numbers add-relation greaterp)
    (worth3 list-of-numbers add-relation equal)
```

could be added to the agenda for later execution.

The worth is an "interestingness" number that is used to order the tasks (the most interesting is tried first). The interestingness of a task is computed on the basis of the intrinsic worth of the concept-to-be-modified (which is available in its worth slot) and the utility of the suggested-heuristic (as specified in its worth slot). The worth values of the concepts in the initial knowledge base are assigned by the programmer. New concepts take their worth values from the worth value of the task that generated them. This can be further reduced if the new concept doesn't have enough examples. In any event the worth of a new concept cannot exceed the worth of the concept from which it was built. This will guarantee that eventually the discovery program will terminate since as time goes on the worth of concepts will tend to 0. Ultimately there will be no interesting tasks left on the agenda.

3. Experimenting with the System

The system has been fully implemented in FranzLisp. In the initial knowledge base, there were 1 operation ("search"), 7 relations ("numberp", "atom", "listp", "oddp", "lessp", "alphaless", and "memq"), 1 structure ("list"), and 26 heuristic rules. The heuristics included the "change-relation", "change-domain", and "add-relation" heuristics mentioned in the last section, and also included a number of other useful heuristics such as "invert-relation" (which could suggest reversing a relation, e.g. to get "greaterp" from "lessp"); "nesting" (which would allow a structure to be further parenthesized, e.g. to get trees from lists); "divide-domain" (which would divide the domain of applicability of an operation into 2 or 3 parts, especially useful for devising binary search and various tree searches); "add-testing" (which could add various boundary condition tests to an operation, e.g. to check for an element being less than the first element or greater than the last element); among many others. There were no heuristics to generate new heuristics in this version of the system.

Many runs were made of the system, fine tuning the initial worth values, adding in new heuristics, and making generalizations to various parts of the program. The final version of the system executed 800 tasks, built 400 frames, and succeeded in keeping 10 relations, 90 structures, and 60 operations which it found interesting (i.e. structures for which is could find examples; operations and relations it could successfully execute). Approximately 20 CPU hours were needed to achieve this level of discovery.

Perhaps the most interesting discovery was "binary-search". A skeletal version of the Isa hierarchy of concepts generated in the discovery of "binary-search" (and the main heuristics used) is shown in Figure 1. The salient steps in the discovery process can be summarized in the following steps:

-The system started with no tasks on the agenda. It added tasks to modify and extend its knowledge base.

-It tried to extend "search" to a new domain, "list".

-The system tried to modify structures. It specified a particular structure "list". It tried to modify "list" by changing a relation in the definition of "list". It changed the predicate "atom" to "numberp" and found a new concept "list-of-numbers".

-After the system had spent some time in extending "search" to both "list" and "list-of-numbers", it tried to modify "list-of-numbers".

-By adding a relation "less" to the "list-of-numbers" definition, it discovered the concept "ascending-ordered-list-of-numbers".

-By extending "search" to the new domain "ascending-ordered-list-of-numbers", the system built a new operation "search-ascending-ordered-list-of-numbers".

-Then, it divided the domain of "search-ascending-ordered-list-of-numbers" into two parts, and built a new operation "search-ordered-list-by-2-divisions".

-It modified the operation "search-ordered-list-by-2-divisions" by adding a testing step so that the new operation tests for the desired element in the first part only if the first element of the second part is bigger than the desired element, i.e. the general idea of binary search.

Here is an abridged version of the "binary-search" frame that was concocted by the system:

```
Name: binary-search (name supplied by system user)
Isa: search-ordered-list-by-2-divisions
   Definition:
      Semantic: (comment must be added by system user)
      Parameters: OBJ,LST
      Code:
        Precondition: OBJ is an atom;
                      LST is an ascending-ordered-list-
                      of-numbers
        Succeed: the first element equals OBJ
        Fail: there are no elements in LST; or
                   OBJ is less than the first element; or
                   the last element is less than OBJ
        Recursive: binary-search for OBJ
                      in first half of list;
                     binary-search for OBJ
                      in second half of list
        Worth: 50
```

A number of other recognizable relations, operations, and structures were discovered. In addition to "binary-search", other new operations generated were "binary-search" (again - by another route), "tree-search", and "ordered-binary-tree-search". Among the relations discovered were "greaterp", "evenp", "alphagreater", "less-than-all" (e.g. 1 is less-than-all (2 6 53 25)), "greater-than-all", "not-memq", and "not-numberp". Interesting structures that were discovered include "binary-list", "ternary-list", "list-of-numbers", "ascending-ordered-list-of-numbers", "descending-ordered-list-of-numbers", "ordered-tree", "binary-tree", "binary-forest", and "forest". The most widely useful heuristics in generating these concepts were "invert", "add-relation", "nesting", "change-domain", and "divide-domain". Figures 2, 3, and 4 show the interesting structures, relations, and operations discovered by the system, including the heuristics used to generate them.

4. Conclusion

The discovery system has discovered a number of relevant concepts, both declarative and procedural, in an important area, data structures. The declarative concepts are learned through relatively straightforward manipulations of lists. The procedural concepts are learned through identifying different parts of a procedure (e.g. precondition step, recursive step, base steps of a recursion, etc.) and manipulating the code for each step separately. The particular heuristics used to generate new concepts from old (e.g. "nesting", "add-relation", "invert-relation", etc.) suggest genetic relationships among the various data structure concepts. It is interesting that so few heuristics could be so useful.

There are still shortcomings to the system. The interestingness criteria are fairly primitive and need to be refined to cut down the number of irrelevant concepts generated. This might be done by more closely tying the data structures generated to the procedures that use them, and call a structure "interesting" only if it is useful to some other procedure. Some of the heuristics are overly specific (e.g. "add-testing", which adds tests to check for something being less than the first element or bigger than the last, seems a bit "special purpose", although the AM system, for instance, has shown the usefulness of testing for extreme cases in the mathematics domain). Moreover, it is difficult to see how certain classes of concepts could be generated by these heuristics. For example, while it seems just possible that with a few additional heuristics something like sorting could be handled, but how could hashing ever be discovered?

This raises an interesting question as to what algorithms and structures can potentially be generated by the current system. The power to modify existing structures and procedures lies primarily in the heuristics, so in order to answer this question the full implications of the heuristics must be understood. Each of the current heuristics has been specifically chosen to lead the system from its initially simple ideas of linear lists and sequential searches to more complex data structures and searches. Certainly, if run further, the system could, with little or no modification, use these heuristics to discover strange new data structures of limited utility (e.g. alternating pairs and triples of numbers, deeply nested structures, etc.) or varieties of sequential or binary search procedures. However, brand new heuristics would be necessary to achieve a breakthrough into substantially different classes of structures, such as arrays, database architectures, networks and their affiliated procedures.

Another aspect of the current approach is the breakdown of the "Code" slot into parts such as pre-condition, termination condition, if-part, then-part, recursive step, etc. Each of these parts is manipulated separately by the heuristics to achieve revised procedural capability. Again, the particular kinds of code parts have been chosen with an eye to producing complex structures and searching algorithms, and may not be fully appropriate in devising other procedures, especially for things like array manipulation which might require code parts appropriate for iteration rather than recursion. Even allowing the addition of different kinds of parts to the code slot in the current system, it would seem extremely difficult to evolve genetically from "list" and "search" to iterative algorithms without additional heuristics and perhaps extra starting concepts as well.

The biggest aid to further extension of the system would be achieving the ability to generate new heuristics automatically. The frame structure for heuristics and the procedure manipulation capabilities should make this possible without undue upheaval to the current architecture. Figuring out exactly what to modify when, though, will still be a difficult task, hopefully one ameliorated somewhat by other work on heuristics (e.g. Lenat [1982a,b], Mitchell [1983], Langley [1983]). Once it can learn new heuristics, the discovery system should be able to learn new things about discovery. As it stands, the system is a promising start towards this goal.

5. Acknowledgements

We would like to thank Marlene Colbourn, Nick Cercone, and Rick Bunt for their input into the thesis on which this paper is based. We would also like to acknowledge the National Sciences and Engineering Research Council of Canada and the University of Saskatchewan for supporting this research both financially and through the provision of computing facilities. Finally, we would like to thank the referees for their constructive comments which helped to make the final version of this paper substantially better than the original version.

6. References

Aref [1983]. M. Aref, Automated Concept Discovery in Data Structure Applications, Technical Report TR-83-10, Dept. of Computational Science, U. of Saskatchewan, Saskatoon, Saskatchewan, August 1983.

IJCAI [1983]. A. Bundy (ed.), Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83), Karlsruhe, West Germany, August 1983.

Langley [1980]. P. Langley, Descriptive Discovery Processes: Experiments in Baconian Science, Technical Report CMU-CS-80-121, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa., May 1980.

Langley [1983]. P. Langley, "Representational Issues in Learning Systems", IEEE Computer, 16, 10, October 1983, pp. 47-51.

Lenat [1977]. D. B. Lenat, "Automated Theory Formation in Mathematics", Proc. IJCAI-77, Cambridge, Mass., August 1977, pp. 837-842.

Lenat [1982a]. D. B. Lenat, "Heuretics: Theoretical and Experimental Study of Heuristic Rules", Proc. AAAI-82, Pittsburgh, Pa., August 1982, pp. 159-163.

Lenat [1982b]. D. B. Lenat, "The Nature of Heuristics", Artificial Intelligence Journal, 19, 2, October 1982.

IMLW [1983]. R. S. Michalski (ed.), Proc. of the International Machine Learning Workshop, Allerton House, Monticello, Illinois, June 1983.

Michalski, Carbonell, and Mitchell [1983]. R. S. Michalski, J. G. Carbonell, T. M. Mitchell (eds.), Machine Learning: An Artificial Intelligence Approach, Tioga, Palo Alto, California, 1983.

Mitchell [1983]. T. M. Mitchell, "Learning and Problem Solving", Proc. IJCAI-83, Karlsruhe, West Germany, August 1983, pp. 1139-1151.

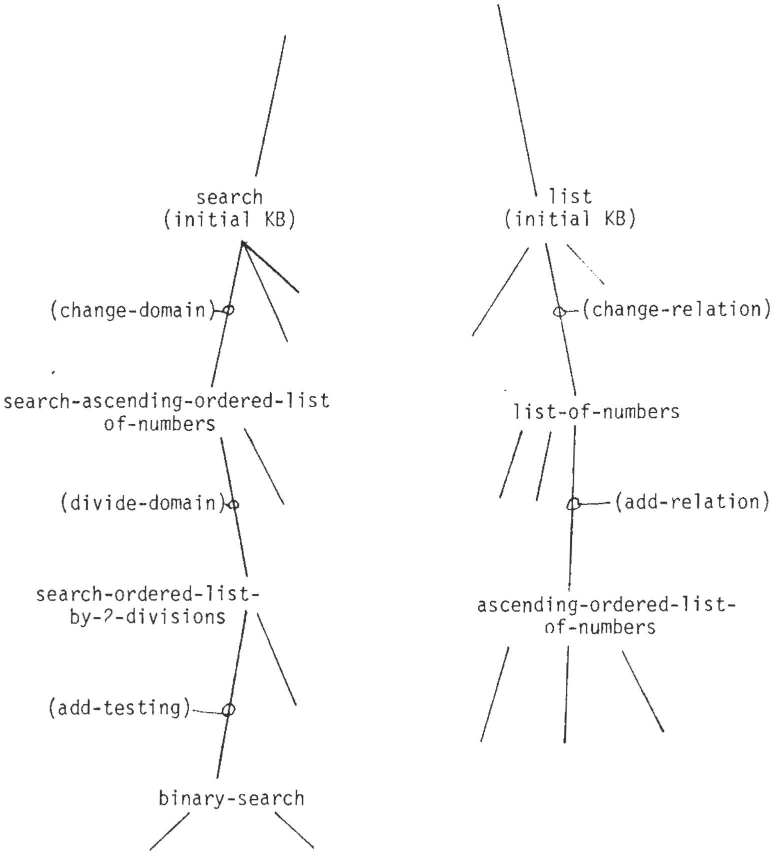


Figure 1 - A Portion of the Isa Hierarchy of Discovered Concepts.
(showing the heuristics which generated them)

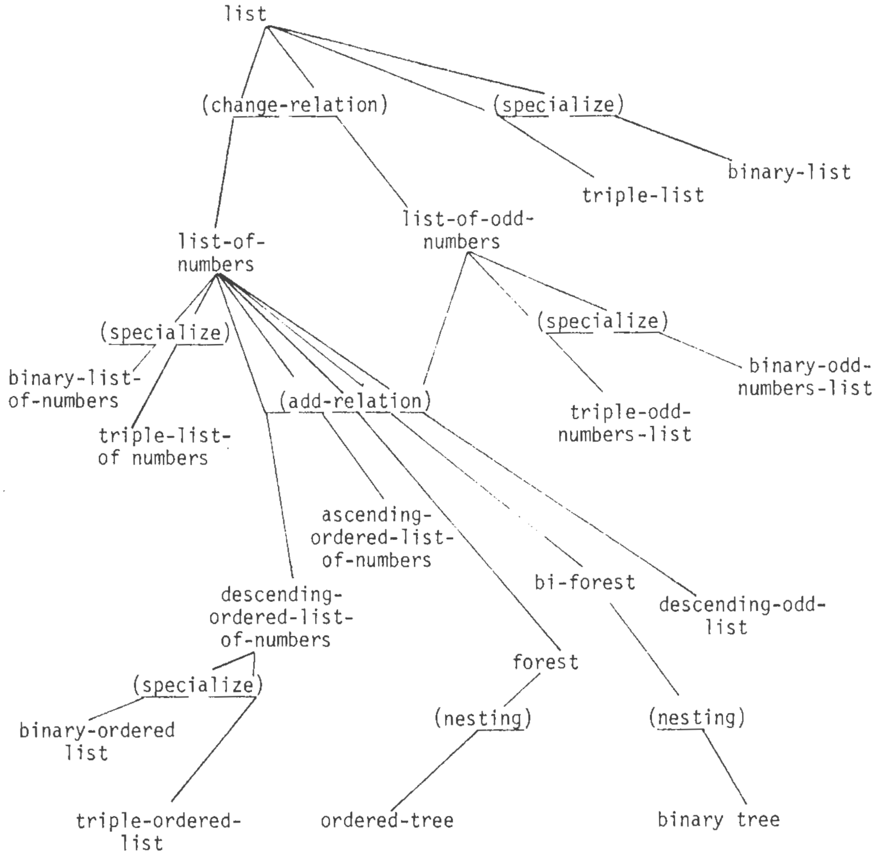


Figure 2 - The Discovered Structures.

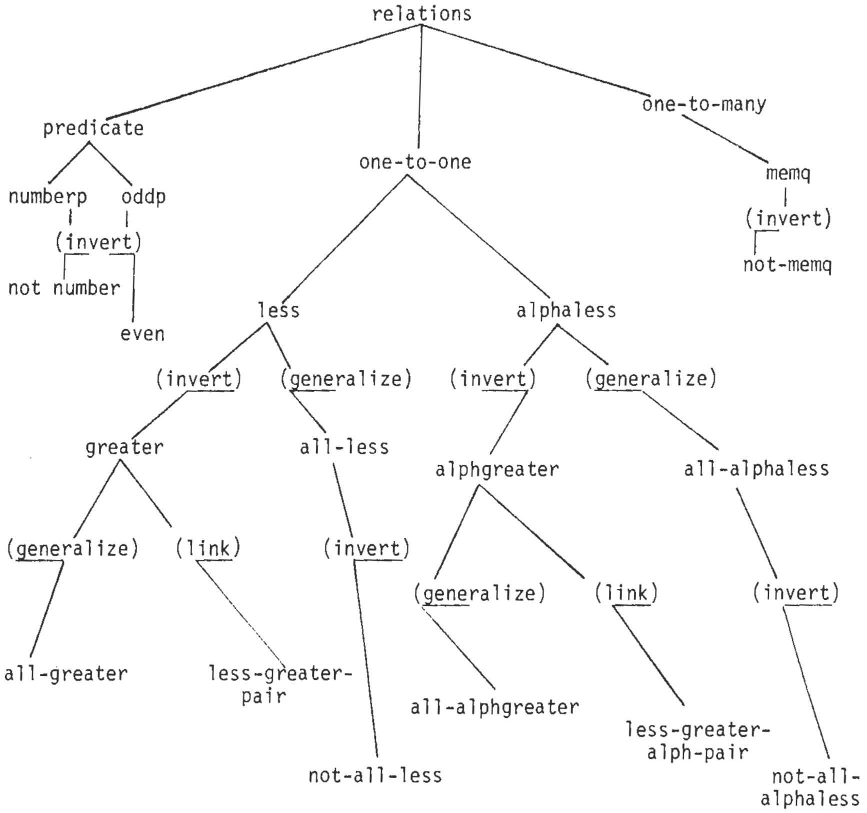


Figure 3 - The Discovered Relations.

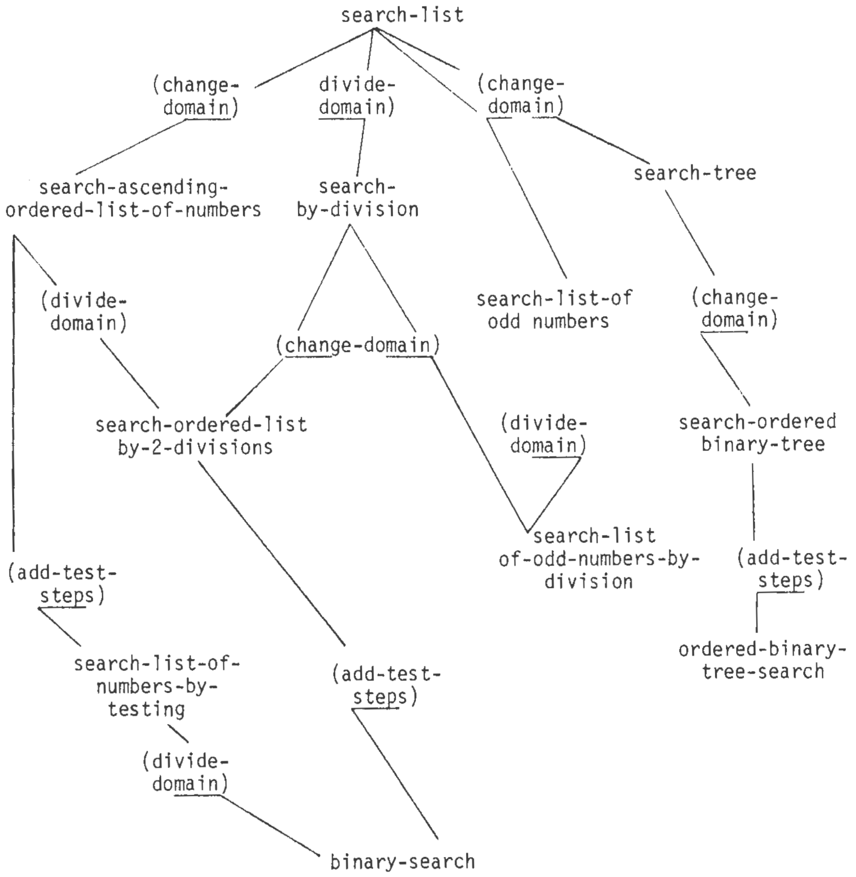

search-list
(change-domain)
divide-domain)
(change-domain)
search-ascending-ordered-list-of-numbers
search-by-division
search-tree
(divide-domain)
(change-domain)
search-list-of odd numbers
(change-domain)
search-ordered-list by-2-divisions
(divide-domain)
search-ordered binary-tree
search-list of-odd-numbers-by-division
(add-test-steps)
(add-test-steps)
search-list-of-numbers-by-testing
(add-test-steps)
ordered-binary-tree-search
(divide-domain)
binary-search

Figure 4 - The Discovered Operations.

THEORY FORMATION AND CONJECTURAL KNOWLEDGE IN KNOWLEDGE BASES

James P. Delgrande

Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4

Abstract

Some fundamental problems concerned with the formation and incorporation of conjectural knowledge in a knowledge base are addressed. From a minimal set of assumptions a logical language for constraining and interrelating a set of consistent hypotheses, or theory, of a domain is derived. It is shown, by means of the formal properties of this language, that this approach allows a basic, yet broad and interesting, set of potential conjectures. Moreover, it is argued that the restoration of consistency of a theory in the face of conflicting evidence may be carried out with reasonable efficiency. Lastly, it is shown that this approach may be incorporated into a well-specified representation system.

Introduction

An important and difficult issue of knowledge representation research concerns the acquisition and incorporation of new knowledge into a knowledge base. Broadly speaking, a knowledge base (kb) may be "told" a statement concerning some domain, or it may "discover" such a statement. Clearly any fully general learning system must be able to independently derive or induce such statements. This paper examines some of the fundamental problems faced by such an autonomous learning system.

The approach taken here is as follows. Initially it is assumed that nothing is known or supposed about the domain, except that it is describable in terms of predicates on individuals; further, nothing is assumed about the underlying knowledge structures of the learning system, except of course that ground instances are representable. Such a system is termed "pure" to distinguish it from other types of learning systems. The only input to such a system then must consist solely of ground instances. On the basis of these instances the system may form general statements, or conjectures, concerning the domain. This set of conjectures is altered only when a conflicting ground instance is encountered. Issues arise concerning what conjectures should be formed and when; and how consistency may be maintained in the presence of conflicting instances.

The early work of John Seely Brown on theory formation [4] is perhaps the closest in spirit to the present work. However a large number of ai systems have dealt with inducing rules or relations from ground instances, but where the domain is assumed governened by an underlying grammar. See [8] for foundational work, and [1] for a recent survey. The field of learning and inductive inference in general is surveyed in [7]. The approach taken here then is to not make such assumptions concerning the domain, and to ignore pragmatic issues dealing with, for example noise, constraints, presuppositions, etc., and instead to address the "lower-level" issues of learning.

The claim here is that the issues faced by pure learning systems are the same as those that must be faced by *any* learning system or knowledge acquisition system (or else, one way or another, explained away). Hence pure systems represent the basic or essential features and principles of learning systems in general. Perhaps somewhat surprisingly, it is not overly difficult to extend a pure system to a broader one, where *a priori* assumptions about the [illegible]

The next section considers the general problem of forming and maintaining a theory of a domain. This is followed by a discussion of specific issues and problems related to introducing conjectures. After this an algebra, thence logic, is presented for expressing a theory of a domain. The formal properties of this logic serve to precisely specify the set of potentially formable conjectures, as well as provide a means of enforcing the consistency of a theory. This work is extended to deal with ground instances where the predicates may now take sets as arguments. Lastly, it is shown how a theory may be mapped into a well-specified knowledge representation system. Further details and extensions of this work may be found in [6].

Theory Formation

A *theory* of a domain will be taken to be simply a consistent set of conjectures concerning a domain. Since all that can be known of a domain by a pure theory formation system are ground instances, all general statements (excepting logical truths) are, of necessity, conjecture.

It is important to distinguish *forming* a theory from *reasoning* within one. The first area deals with the issues involved in introducing conjectures and maintaining the consistency of a theory. Since in forming conjectures one "goes beyond" the facts at hand, this activity is largely non-deductive. Reasoning within a theory on the other hand is a primarily deductive activity; much of current representation research has been concerned with what inferences may be drawn, how these inferences may be drawn, and how knowledge may be structured to best facilitate these inferences. There is no reason then why a system that is well-suited to performing deductions from a particular theory should also be well suited to altering the contents of the theory. Thus, for example, it is one thing to use a theory to predict the motion of known planets and the existence of unseen planets; it is quite another to formulate a theory about how planets move.

One possibility for representing conjectures is to use *prototypes* [11], [15]. In such a case membership in the extension of a term is a graded affair, and is a matter of similarity to a representative member, or prototype. Thus one might say that it is more "ravenlike" for a particular raven to be black, rather than some other colour. I wish however to explicitly reject this approach, and instead deal with universal statements which represent (hypothesised) laws governing a domain. Thus one would make the stronger claim that "all ravens are black". The advantage of this approach is that it allows a much greater degree of overall kb systematisation through the interrelation of predicates, as well as permitting a full logical apparatus for reasoning with conjectures. The problem with this latter approach, of course, is that one rarely, if ever, naturally encounters exceptionless laws. Thus the above generalisation is "falsified" by the existence of an albino raven; failing that, some malcontent is always likely to paint a raven red.

The problems of rationalising and allowing exceptions is beyond the scope of this paper; again the reader is referred to [6]. Suffice to say however that in verifying a scientific law, for example "water boils at 100°C", one must rely on a host of underlying assumptions, such as the pressure is 760 mm., the water is pure, the thermometer accurate, etc. Such assumptions then may be used, even if unknown, to excuse an exception.

It is instructive also to specify the properties that conjectures should have. To begin with, before a conjecture is formed, there should be a reason for so doing, and thus *evidence* in support of it. Evidence then must also discriminate among the other feasible hypotheses. On the other hand a falsified conjecture should (subject to the above caveat concerning exceptions) be abandoned. Also the properties of a conjecture should be the same, wherever possible, as the corresponding "known" formula. Thus, for example, if P and Q were hypothesised to be equal, we would like this hypothesised equality to behave similarly to standard extensional equality.

Clearly the set of conjectures must be consistent with what is both known and conjectured to be true. Also, while a conjecture can never be demonstrated to be true, the set of conjectures should converge to the "true" solution, or identify the underlying relations "in the limit" [8]. Note though that for reasons of efficiency we don't want to know all that potentially may be known -- that is the truth values of all known predicates applied to all known individuals.

Introducing Conjectures

Corresponding to each known predicate symbol P in the domain, we will have a set of individuals that P is known to be true of, and a second set which it is known to be false of. Hence for each known P we can specify sets P_+ and P_- by:

Define: $P_+ = \{a \mid K\ P(a)\}$ $P_- = \{a \mid K\ \neg P(a)\}$

The operator K is as in [9] and may be read as "it is known that". Thus all information known about P is contained in P_+ and P_-, and can say:

Define: $P = (P_+, P_-)$.

These notions clearly generalise to higher-place predicates.

To make the notation clear, upper case Roman letters will be used for the names of standard, set-theoretic predicates, while upper case italic letters will be used to stand for what is known about the corresponding predicate, viz. the sets P_+ and P_-. Furthermore, the hypothesised relation and operation signs are obtained by subscripting an "h" to the corresponding set-theoretic signs. Thus, for example, while P=Q will mean that P and Q are (extensionally) equivalent, $P =_h Q$ will mean that P and Q are hypothesised to be equivalent.

Now, we can conjecture that P and Q are equal when

i) there is some evidence to do so (i.e. $P_+ \cap Q_+ \neq \phi$) and

ii) it is not known that P and Q are not equal (i.e. $P_+ \cap Q_- = \phi$ and $P_- \cap Q_+ = \phi$).

Continuing in the same manner, we get the following conditions for forming conjectures.

Define:

$$P =_h Q \text{ iff } P_+ \cap Q_+ \neq \phi \wedge P_+ \cap Q_- = \phi \wedge P_- \cap Q_+ = \phi$$
$$P \subset_h Q \text{ iff } P_+ \cap Q_+ \neq \phi \wedge P_+ \cap Q_- = \phi \wedge P_- \cap Q_+ \neq \phi$$
$$P \subseteq_h Q \text{ iff } P_+ \cap Q_+ \neq \phi \wedge P_+ \cap Q_- = \phi$$
$$P\,!_h\,Q \text{ iff } P_+ \cap Q_+ = \phi \wedge P_+ \cap Q_- \neq \phi \wedge P_- \cap Q_+ \neq \phi$$

The condition $P_- \cap_h Q_- \neq \phi$ is not included because, assuming an arbitrarily large universe of potentially knowable individuals, it tells us nothing about the relationship between P and Q. Clearly though any coherent approach to forming conjectural relations must at least include the above conditions. However these definitions lead immediately to problems. Consider, for example, the case where all that is known are instances P(a), Q(a), Q(b) and R(b). Certainly $P =_h Q$ and $Q =_h R$, but we cannot form the hypothesis $P =_h R$. The problem is that according to the above definition there is no direct evidence for $P =_h R$. The only way out of this dilemma (short of maintaining separate kb's for the different alternatives) is to test one of the known individuals against the predicates in question so that evidence for the hypothesis may be obtained. Thus, in the above example, if we determine that P(b) is true, we can assert $P =_h R$. If P(b) turns out to be true on the other hand, we have falsified $P =_h Q$ and can replace it by $P \subset_h Q$. In this latter case the conjectures $P \subset_h Q$ and $Q =_h R$ should yield $P \subset_h R$. Determining the truth value of R(a) allows one of either $P \subset_h R$, or $P\,!_h\,R$ (and $Q \subset_h R$) to be asserted. The fact that individuals can always be located from among the set of known individuals to restore consistency in this manner follows from the results of the next section.

There are however pragmatic as well as formal considerations in forming conjectures. That is, one might be somewhat nervous hypothesising, for example, that all ravens are black (or, more starkly, all A's are B's) on the basis of a single observed individual, and might prefer more (say, 42) of such common individuals. In this paper though I am concerned strictly with the formal aspects of such conjectures and thus pragmatic considerations have no bearing on the issues discussed here.

The notions of hypothesised complement, relative complement, union and intersection are easily and fairly intuitively introduced.

Define:

$$\neg_h P = (P_-, P_+)$$
$$P -_h Q = (P_+ \cap Q_-, P_- \cup Q_+) \qquad (= P \cap_h \neg_h Q)$$
$$P \cup_h Q = (P_+ \cup Q_+, P_- \cap Q_-)$$
$$P \cap_h Q = (P_+ \cap Q_+, P_- \cup Q_-)$$

Thus we can form conjectures such as

$Father\_of\ !_h\ Mother\_of$

$Raven \subset_h Flies$

$Bachelor =_h Male \cap_h Unmarried$.

The language corresponding to the set of sentences which may be formed under standard rules of composition from the above operations is called EL.

These operations for the most part behave analogously to their set-theoretic counterparts. For example, the associative and commutative laws hold, as do the distributive laws and De Morgan's laws. However, relations involving the hypothesised complement are problematic, and many desirable relations do not work out as expected. For example, neither

$$(P \subset_h Q) \supset (\neg_h Q \subset_h \neg_h P) \quad \text{nor} \quad P =_h \neg_h \neg_h P$$

hold in general. The reason for these difficulties is that while shared positive instances are required to hypothesise equality or containment relations, we would require shared negative instances when considering the negations of predicates. These problems disappear then if we add $P_- \cap Q_- \neq \phi$ to the definitions of the hypothesised relations. However, while this fix would indeed give us a complement that corresponded more closely with the set-theoretic idea of complement, it is entirely unsatisfactory in other respects. First, it violates the requirement for evidence set out previously, that it serve to discriminate among the competing hypotheses. Also, in order to claim that two predicates are disjoint, or that one contains the other, it is totally unreasonable to require that they share a common negative instance, as the above solution would demand.

The resolution to this problem is to simply accept things as they are. First, the idea of complement in set theory is a not entirely reputable notion [10]. Second, the conjectures assumedly concern natural kind concepts, and there is good reason to believe that the complement of such a concept is not itself a natural kind [12]. Lastly, a shared negative instance for two predicates does not tell anything about how their extensions are related. These considerations are put on a firmer, formal footing in the next section.

A Logic for Conjectures

The central ideas of this section are as follows. Expressions formed from the hypothesised operations satisfy many standard algebraic properties. Thus, for example, we are guaranteed that $\alpha \cap_h \beta = \beta \cap_h \alpha$, for suitably defined "=". The algebra, called HL_a, corresponding to this system is investigated. By introducing an operation analogous to implication, a corresponding propositional logic, HL_i, is derived. This logic can be used to guide hypothesis formation in HL and to enforce consistency.

Thus we have:

Define: If α and β are expressions formed from known predicate names, using the operations $\cap_h$, $\cup_h$, $\neg_h$, and $-_h$:

$$\alpha = \beta \text{ iff } \alpha_+ = \beta_+ \text{ and } \alpha_- = \beta_-$$
$$\alpha \leq \beta \text{ iff } \alpha_+ \subseteq \beta_+ \text{ and } \beta_- \subseteq \alpha_-$$
$$\alpha < \beta \text{ iff } \alpha \leq \beta \text{ but } \alpha \neq \beta$$

Define: If I is the set of known individuals then lower and upper bounds for the expressions of HL are defined by:

$$\mathbf{0} =_{df} (\phi, \mathrm{I}) \qquad 1 =_{df} (\mathrm{I}, \phi)$$

Following from this we obtain:

Theorem: If α, β, and γ are expressions of HL, then:

$\alpha \cap_h \beta = \beta \cap_h \alpha \quad \alpha \cup_h \beta = \beta \cup_h \alpha$	Commutative
$\alpha \cap_h (\beta \cap_h \gamma) = (\alpha \cap_h \beta) \cap_h \gamma$	Associative
$\alpha \cup_h (\beta \cup_h \gamma) = (\alpha \cup_h \beta) \cup_h \gamma$	
$\alpha \cap_h (\alpha \cup_h \beta) = \alpha \quad \alpha \cup_h (\alpha \cap_h \beta) = \alpha$	Absorption
$\alpha \cap_h (\beta \cup_h \gamma) = (\alpha \cap_h \beta) \cup_h (\alpha \cap_h \gamma)$	Distributive
$\alpha \cup_h (\beta \cap_h \gamma) = (\alpha \cup_h \beta) \cap_h (\alpha \cup_h \gamma)$	
$\alpha \cap_h \alpha = \alpha \quad \alpha \cup_h \alpha = \alpha$	Idempotent
$\alpha \cap_h (\beta \cup_h (\alpha \cap_h \gamma)) = (\alpha \cap_h \beta) \cup_h (\alpha \cap_h \gamma)$	Modularity
$\alpha \cup_h (\beta \cap_h (\alpha \cup_h \gamma)) = (\alpha \cup_h \beta) \cap_h (\alpha \cup_h \gamma)$	
$\alpha \cap_h \mathbf{0} = \mathbf{0} \quad \alpha \cup_h \mathbf{0} = \alpha$	Univ Bds
$\alpha \cap_h 1 = \alpha \quad \alpha \cup_h 1 = 1$	
$\alpha = \neg_h \neg_h \alpha$	Involution
$\neg_h (\alpha \cup_h \beta) = \neg_h \alpha \cap_h \neg_h \beta$	De Morgan
$\neg_h (\alpha \cap_h \beta) = \neg_h \alpha \cup_h \neg_h \beta$	

This algebra is nearly, but not quite, a Boolean algebra. For it to be a Boolean algebra, we would also require the existence of a complement, $\neg_h \alpha$, for each element α, where $\alpha \cap_h \neg_h \alpha = \mathbf{0}$ and $\alpha \cup_h \neg_h \alpha = 1$. However, no such element (provably) is possible here in general. The system that we do obtain instead has been called a De Morgan lattice [2] or a quasi-Boolean algebra [13].

A ply operator, $\supset$, which will correspond to implication in HL_i is bound by the following postulates [5]:

$$\alpha \cap_h (\alpha \supset \beta) \leq \beta$$
$$\text{if } \alpha \cap_h \gamma \leq \beta \text{ then } \gamma \leq (\alpha \supset \beta).$$

The first postulate corresponds to modus ponens, while the second says that the ply is maximal among solutions to the first.

Theorem: $\alpha \supset \beta = (\mathbf{I} - ((\alpha_{+} - \beta_{+}) \cup (\beta_{-} - \alpha_{-})), \beta_{-} - \alpha_{-})$

Corresponding to this algebra, the following logic is derived:

Axioms:

A1 $\alpha \supset (\beta \supset \alpha)$

A2 $(\alpha \supset (\beta \supset \gamma)) \supset ((\alpha \supset \beta) \supset (\alpha \supset \gamma))$

A3 $\alpha \wedge \beta \supset \alpha$

A4 $\alpha \wedge \beta \supset \beta$

A5 $\alpha \supset (\beta \supset (\alpha \wedge \beta))$

A6 $\alpha \supset (\alpha \vee \beta)$

A7 $\beta \supset (\alpha \vee \beta)$

A8 $(\alpha \supset \gamma) \supset ((\beta \supset \gamma) \supset (\alpha \vee \beta \supset \gamma))$

A9 $\alpha = \neg \neg \alpha$.

Rules of Inference:

MP From $\vdash \alpha$ and $\vdash \alpha \supset \beta$ infer $\vdash \beta$.

NH $\vdash \alpha \supset \beta$ iff $\vdash \neg \beta \supset \neg \alpha$.

The propositional connectives $\wedge$, $\vee$, $\supset$ and $\neg$ of HL_l correspond to the operations $\cap_h$, $\cup_h$, $\supset$ and $\neg_h$ in HL_a. The sign '$\vdash$' is given its standard definition for provability, while '$\Vdash$' can be defined by:

Define: $\Vdash \alpha$ iff $\alpha' = 1$, where α' is the corresponding formula in HL_a.

Theorem: $\vdash \alpha$ iff $\Vdash \alpha$.

Two proofs of this theorem have been obtained. The first links the logic HL_l with its (Lindenbaum) algebra HL_a. The second treats HL_l as a three-valued logic (with intermediate truth value "unknown"). Truth tables for the logic are obtained, and tautologies are shown to be theorems of HL_l (and vice versa). From this we get the easy result:

Corollary: HL_l is decidable.

Lastly we get:

Theorem: If $\{\alpha_1, \cdots, \alpha_n\} \vdash \beta$ in HL_l and $\alpha_1', \cdots, \alpha_n', \beta'$ are the corresponding expressions of HL, and $\alpha_1', \cdots, \alpha_n'$ have been hypothesised, then known individuals may be located which will either

i) provide evidence to allow β' to be hypothesised, or

ii) falsify one of $\alpha_1', \cdots, \alpha_n'$.

Moreover, the proof of this theorem shows how such individuals can be located in O(n) time. Thus while only a small proportion of the potentially knowable ground instances typically are known, the above result guarantees that instances can readily be found to resolve any inconsistencies.

Thus we can form and enforce the consistency of a set of conjectures which correspond to the sentences of a Boolean algebra, except that we cannot refer to absolute complements. Translated into propositional logic, this means that we lose the law of the excluded middle and proof by contradiction.

This logic provides a rather nice solution to the so-called paradox of confirmation [3]. The paradox arises from the claim that whatever evidence supports a generalisation also supports all logical consequences of the generalisation. Thus, for example a non-black non-raven supports $\neg$*Black* $(x) \supset \neg$*Raven* (x) and thus *Raven* $(x) \supset$ *Black* (x). This means that a non-black non-raven (e.g. this paper, or Jupiter) provides evidence for the assertion that all ravens are black.

The paradox is resolved here by replacing

$\vdash (\alpha \supset \beta) \equiv (\neg \beta \supset \neg \alpha)$ by

$\vdash \alpha \supset \beta$ iff $\vdash \neg \beta \supset \neg \alpha$.

Thus, if it is the case that all ravens are indeed black, then it is the case also for the contrapositive.

Further Results

This section briefly summarises additional work that has been carried out. This work breaks down into two subcategories: the set of individuals over which predicates may range is extended to include sets, and the set of (hypothesised) operations is expanded.

In the first case, in addition to ground instances of the form *raven* (a) or *above* (a, b), instances such as *stack* $(\{a_1, a_2, a_3\})$, or *stack* (S) where S is defined to be $\{a_1, a_2, a_3\}$, are permitted. This however leaves us with two types of sets (or, perhaps, "set-like objects"). First there are the finite, known sets which may now appear as predicate arguments. These include not only collections of primitive individuals, like $\{a_1, a_2, a_3\}$, but also may include collections of predicate names, for example {*Red* , *Orange* , *Green* }. These sets are called ***reducible***. Second are the predicate extensions. These may be either finite or infinite; however neither this, nor the extensions themselves, can ever be known. All that can be known is (from before) the subsets of known individuals that the predicate is and is not known to be true of. This is expressed by, for example, $Red = (Red_{+}, Red_{-})$. Such sets are called *irreducible*.

The set of allowable reducible and irreducible sets, and the interaction between the two types, are specified by modifying the Zermelo-Fraenkel axiomatic set theory. However, "higher-level" axioms, including the axiom of infinity, are not included. This gives us an infinite universe of knowable individuals, but where no set has an infinite number of known members. In addition, this expanded system is still decidable. (The system actually remains decidable even with an axiom of infinity. However, it just seemed more convenient and natural to limit the universe to "hereditarily finite" sets. Additional axioms could, of course, be added to extend the system.)

An immediate benefit of this extension is that it allows "meta-conjectures" about sets of conjectures to be phrased. Thus, for example, we can express the conjecture "for all types of bears, all individuals of a particular type are coloured the same". Thus

$$x \in Bear\_type \supset (\exists y\ (y \in Colour\_type \wedge (z)(z \in x \supset z \in y))).$$

It is worth noting though tbat if we assume that the only kinds of bears are the known ones, then the above can be expressed in first order terms, and thus all that we've gained is a somewhat more compact notation and higher degree of kb systematisation.

Hypothetical set operations corresponding to the standard domain, range, converse, image and composition operations are also defined for binary predicates (as well as obvious extensions to higher-place predicates). Furthermore, notions corresponding to the transitive closure of a relation, and sets closed under a particular binary relation are also defined. This however doesn't alter the relations that may be expressed; and thus the results concerning HL in the previous section still apply. However, it does increase the individuals that may be referred to. Thus we can now form conjectures such as

$$On \subseteq On^{*} \subseteq_{h} Above \quad \text{or} \quad Stack =_{h} On^{\alpha}$$

where On^{*} is the transitive closure of the On relation, and On^{α} is true of a set of objects closed under the On relation. Thus the above expresses the facts that if something is known to be *On* something else, then these objects are (known to be) in the transitive closure of *On*: if a pair of objects are linked by an *On* chain, then the first is hypothesised to be *Above* the other; and a stack of blocks is hypothesised to be such that every pair of blocks is in the transitive closure of *On*.

I have not examined the computational problems involved in forming and maintaining the consistency of such conjectures. Unlike the conjectures of the previous section, it is not obvious how efficient or inefficient such procedures would be, since the set of formable individuals has been vastly expanded. However, here as in the previous cases, decidability is retained.

Reasoning with Conjectural Knowledge

The previous sections dealt with the problems of constructing a theory of a domain; this section summarises the issues and approaches taken in reasoning within a theory. Basically what we want to do is take a theory (which is entirely conjecture) and affix it to a kb which contains general and arbitrary sentences.

For this, I have taken an existent language, KL [9] and extended it to allow sentences that are only hypothesised to be true; the resulting language is called HKL. In brief, KL is a logical language that can refer both to application domains and to what knowledge bases might know about such domains. This was accomplished by adding a sentential operator, K, to first order predicate calculus, where Kα could be read as "α is known to be true". A semantic and proof-theoretic analysis was provided for the language (KL) as well as for operations of answering queries, acquiring knowledge and assuming defaults.

The basic idea with HKL is that KL is extended by the addition of a sentential operator, H, where Hα can be read as "α is conjectured to be true". Thus a distinction is made between what is strictly known (K) and what is only surmise (H). The semantic and proof-theoretic analyses of KL are extended appropriately for HKL. Following from this, it is straightforward to specify a translation of sentences of HKL into KL. A problem does arise however with conjectures that quantify over predicates. This may be avoided (but perhaps not entirely satisfactorily) by assuming that the only predicates quantified over are the known predicates.

HKL also allows a minor extension to default theories of reasoning. A sentence such as 'typically birds fly" might be expressed in HKL as

$$(\mathrm{Bird}(x) \wedge \neg \mathrm{K} \neg \nabla \mathrm{Bird}{:}i\,(x)) \supset \nabla \mathrm{Bird}{:}i\,(x)$$

$$\nabla \mathrm{Bird}{:}i\,(x) \supset \mathrm{Fly}(x)$$

where ∇Bird:i is interpreted as "x inherits the i-th default property of Bird", or, in Reiter's notation [14], as

$$\frac{\mathrm{Bird}(x): \mathrm{Fly}(x)}{\mathrm{Fly}(x)}$$

However, if one knows that something is a bird, and it is consistent to believe that it flies, the best that one can really do is *hypothesise* that it flies. Thus:

$$(\mathrm{Bird}(x) \wedge \neg \mathrm{K} \neg \nabla \mathrm{Bird}{:}i\,(x)) \supset \mathrm{H}(\nabla \mathrm{Bird}{:}i\,(x))$$

or

$$\frac{\mathrm{Bird}(x): \mathrm{Fly}(x)}{\mathrm{H}(\mathrm{Fly}(x))}$$

Use of the operator H then is sanctioned through HKL.

Conclusion

This paper has presented an approach to autonomous learning. The major characteristic of this approach is that nothing is assumed concerning the domain or the underlying representation scheme, except that the former is assumed describable in terms of predicates and individuals, and that the latter can represent these instances. The predicates are taken as first ranging over (primitive) individuals, and later over finite sets.

Conjectural relations may be formed according to elementary evidence conditions, and new hypothetical sets may be created using the set of hypothetical operations. The underlying logic, HL_l, is used to ensure the consistency of a set of conjectures. Also the problem of restoring consistency in the face of a conflicting instance is quite efficient. This logic also precisely specifies the set of legal formable conjectures; an axiomatic approach to the entities of the system -- the primitive individuals and reducible and irreducible sets -- also precisely specifies the set of legal individuals.

A number of important open problems remain. Certainly if the system is to be implemented for noisy, real world situations, then issues concerned with such situations need to be dealt with. At present the problem of exceptions is being addressed: that is, when should exceptions be admitted, and, more importantly, what it means for something to be an exception and how this affects the meaning of the violated rule. Also the matter of definition needs to be addressed. Given a collection of conjectures concerning a set of predicates, we need to be able to specify a subset of the predicates as being primitive, and define the remaining predicates in terms of these. This systematising of the kb also leads to a consideration of the introduction of new terms. Thus, for example, it would be useful to be able to conjecture that the class of people consists of two (unknown) subclasses (say, "male" and "female") based on observed symmetries of kinship relations.

Since this approach deals explicitly with sets and their interrelations, it may provide a means for investigating learning with respect to those schemes that deal explicitly with classes and concepts, namely semantic nets. This would be in direct contrast with most other systems, which are concerned with learning production rules, or other such condition/action pairs.

References

[1] D. Angluin and C.H. Smith, "A Survey of Inductive Inference: Theory and Methods", Technical Report 250, Department of Computer Science, Yale University, 1982

[2] R. Balbes and P. Dwinger, *Distributive Lattices*, University of Missouri Press, 1974

[3] M. Black, "Notes on the 'Paradoxes of Confirmation'", in *Aspects of Inductive Logic*, Hintikka and Suppes eds., North-Holland, 1966

[4] J.S. Brown, "Steps Toward Automatic Theory Formation" *Proc. IJCAI-73*, 1973, pp 121-29

[5] H.B. Curry, *Foundations of Mathematical Logic*, McGraw-Hill, 1963

[6] J.P. Delgrande, Ph.D. thesis (in preparation), Department of Computer Science, University of Toronto, 1984

[7] T.G. Dietterich, B. London, K. Clarkson and G. Dromey, "Learning and Inductive Inference", report No. STAN-CS-82-913, Department of Computer Science, Stanford University, May 1982 and Chapter XIV of *The Handbook of Artificial Intelligence*, P.R. Cohen and E.A.

Feigenbaum eds., William Kaufmann Inc., 1982

[8] E.M. Gold, "Language Identification in the Limit", *Information and Control 10*, 1967, pp 447-474

[9] H.J. Levesque, "A Formal Treatment of Incomplete Knowledge Bases", Ph.D. thesis, Department of Computer Science, University of Toronto, 1981

[10] J.D. McCawley, *Everything that Linguists have Always Wanted to Know about Logic* *but were ashamed to ask*, The University of Chicago Press, 1981

[11] D.N. Osherson and E.E. Smith, "On the Adequacy of Prototype Theory as a Theory of Concepts", *Cognition 9*, 1981, pp 35-58

[12] W.V.O. Quine, "Natural Kinds", in *Naming, Necessity and Natural Kinds*, S.P. Schwartz ed., Cornell University Press, 1977, pp 155-175

[13] H. Rasiowa, *An Algebraic Approach to Non-Classical Logics*, North-Holland Pub. Co., 1974

[14] R. Reiter, "On Reasoning by Default" *TINLAP-2*, 1978, pp 210-18

[15] E. Rosch, "Principles of Categorisation" in *Cognition and Categorisation*, E. Rosch and B.B. Lloyds eds., Lawrence Erlbaum Associates, 1978

CONCEPTUAL CLUSTERING AS DISCRIMINATION LEARNING

Pat Langley
Stephanie Sage
The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213 USA

ABSTRACT

In this paper we examine the relation between learning from examples and the task of *conceptual clustering*. We briefly review Michalski and Stepp's work in this area, and present an alternate approach based on Hunt and Quinlan's method for learning discrimination networks from examples. The new approach treats all observed objects as positive instances, while unobserved objects are treated as negative instances, with the latter being enumerated from known attributes and their values. This clustering method is more elegant than earlier ones, and generates simple classification schemes that predict the observed objects but none of the unobserved ones.

INTRODUCTION

The ability to acquire new concepts is an essential aspect of intelligence, and the task of learning concepts from examples has been extensively studied by researchers in machine learning. In this paradigm, a tutor presents the learning system with a set of positive and negative instances of some concept (such as *arch* or *chair*). In turn, the system is expected to form some description of the concept that allows it to correctly classify future instances as well as the presented ones.

However, there are many situations in which one must learn concepts without the aid of a tutor, or without explicit feedback of any kind. For instance, children acquire concepts like *chair* or *dog* before they know the associated words, and scientists formulate taxonomic schemes based on observational data. In fact, biologists and statisticians have developed the techniques of cluster analysis and numerical taxonomy [1] to aid in this process, and one can view these methods as automating the process of concept formation. Such techniques accept a set of objects with associated attribute-value pairs as input, and produce a hierarchical classification scheme as output, with similar objects being placed in the same classes.

One disadvantage of traditional clustering methods is that they define classes extensionally – by giving a list of members – rather than intensionally – by giving some description or rule used to assign class membership. In response to this limitation, Michalski and Stepp [2] have explored the process of *conceptual* clustering, in which intensional descriptions are generated along with extensional definitions. Below we briefly review their earlier work in this challenging domain. In the remainder of the paper, we present an approach to conceptual clustering that is quite different from the one described by Michalski and Stepp. We describe the method in terms of search through a space of classification trees, examine its behavior on two clustering tasks, and consider both the advantages and limitations of the approach.

THE CONCEPTUAL CLUSTERING TASK

During the task of learning concepts from examples, explicit feedback is provided with each sample description. In other words, the input is divided into a set of *positive* instances which exemplify the concept to be learned, and a set of *negative* instances or counter-examples to the concept. In contrast, a conceptual clustering system is simply given a set of objects and asked to group similar ones together. However, more subtle differences also exist between the two tasks. First, much of the work on learning from examples has focused on *conjunctive* concepts, in which a conjunction of features or relations defines the concept. In contrast, if one views the classification tree produced during conceptual clustering as the "concept" to be learned, then this concept is always *disjunctive*, since mutually exclusive branches are involved. Second, learning from examples usually requires a *single level* concept to be learned, with no sub-concepts being involved. However, the classification hierarchies generated during conceptual clustering often involve *multiple levels* of description.

Despite the greater complexity of the conceptual clustering task, there are also many similarities to learning from examples, and it is natural to look for ways to apply methods from this domain to the problem of conceptual clustering. In fact, this is precisely the approach taken by Michalski and Stepp with their CLUSTER/2 system [2]. This program employed a method for learning concepts from examples to determine the branches (or concepts) at each level in the classification tree, starting at the top and working downward. In order to do this, it required sets of positive and negative instances, and these were inferred in the following manner. Given the goal of dividing the observed objects into N disjoint classes, CLUSTER/2 randomly selected N *seed* objects. The system treated each such seed as a positive instance of some concept to be learned, and treated other seeds as negative instances. In this way, it arrived at a set of descriptions, each covering a different seed (but none of the other seeds). In addition, each description also covered a number of other (non-seed) observations, and these were assigned to the same class.

Based on these descriptions, CLUSTER/2 produced a new set of seeds representing the central tendency of each description; that is, objects with values occurring in the center of the acceptable range were chosen as the new seeds. Using these as input to the method for learning from examples, the system repeated the process described above, generating a revised set of descriptions. This strategy continued until the seed objects stabilized, giving an optimal set of N disjoint classes, which became branches on the classification tree. The system repeated this process for different values of N, retaining the best classification tree according to some user-specified criterion. CLUSTER/2 then applied the entire process recursively to each subset of objects, adding lower level branches to the classification tree. When finished, the system produced not only a hierarchical clustering of the objects in terms of classes and subclasses, but the rules it had used in making that clustering. In summary, Michalski and Stepp's system employed a method for learning from examples as a subroutine, using it to formulate decision rules at each level in the classification hierarchy.

Other approaches to the conceptual clustering task are possible. For instance, Lebowitz [3] has reported a concept formation system that learns in an incremental fashion, rather than requiring all data at the outset. One can also imagine starting with specific subclasses, and working upwards to more general superclasses. Wolff [4] has described such a "bottom-up" system that operates in the domain of grammar learning, while Langley, Zytkow, Bradshaw, and Simon [5] have described a similar system that works in the domain of chemistry. Since each of these systems allow overlapping classes rather than requiring disjoint groups, D. Fisher (personal communication) has suggested they be called conceptual *clumping* systems rather than clustering systems. In this paper, we will limit our attention to the problem of forming disjoint classification schemes. Now that we have considered Michalski and Stepp's approach to this task, let us turn to another method that relies on a somewhat different mapping to learning from examples.

Table 1. Four objects and their descriptions.

NAME	COLOR	SHAPE	SIZE
OBJECT-1	BLACK	CIRCLE	SMALL
OBJECT-2	BLACK	SQUARE	MEDIUM
OBJECT-3	WHITE	CIRCLE	MEDIUM
OBJECT-4	WHITE	TRIANGLE	LARGE

CONSTRUCTING DISCRIMINATION NETWORKS

Some of the earliest work on learning from examples was carried out by Hunt, Marin, and Stone [6]. In their approach, a concept was viewed as a discrimination network for predicting positive and negative instances, and the concept learning task consisted of constructing this discrimination net. Their system (named CLS) input two sets – one containing positive instances and one containing negative instances – with each instance described as a conjunction of attribute-value pairs. The program began by finding the most discriminating attribute (according to some evaluation function), and created separate branches for each of its values. CLS sorted instances down these branches, and applied the method recursively to each of the resulting subsets, generating new branches and further discriminations. This process continued until every terminal node in the tree contained either all positive instances or all negative instances.

Despite its early occurrence, this approach was largely ignored by later machine learning researchers. There were probably two reasons for this abandonment: attention shifted from all-at-once learning methods to incremental ones; and attention shifted from attribute-value based concepts to relational ones. Hunt's method did not appear to generalize to either of these more difficult cases (though we shall see that this is not quite true for the relational issue). Despite these limitations, the approach has a number of advantages: it requires very little search; it can acquire disjunctive concepts; and it can be easily modified to deal with noise. Quinlan [7] has continued to work within Hunt's original paradigm, using a different evaluation function for directing search through the space of discrimination nets, and implementing a sampling approach that can deal with very large amounts of data.

Hunt and Quinlan's approach to learning from examples is significant to conceptual clustering for a simple reason: the discrimination networks generated by the approach bear a striking resemblance to the classification trees generated by conceptual clustering methods. In addition, the method represents concepts in terms of a number of "levels", and when disjuncts occur, it generates trees with multiple branches at each level. In fact, the only difficulty in using the method for conceptual clustering is the need for a division of data into positive and negative instances, and this is easily corrected. In the conceptual clustering paradigm, only some of the *possible* objects are actually observed, and these can be treated as if they were positive instances. If the possible values of each attribute are known, then it is simple to enumerate those objects which were not observed, and to interpret them as negative instances. Given these two sets, one can construct a discrimination network using Hunt and Quinlan's method. Only a single step is necessary to transform this network into an acceptable classification tree, as we shall see shortly.

The relation between the two tasks can be clarified with an example. Table 1 presents descriptions of four objects in terms of three attributes – **color** (black or white), **shape** (circle, triangle, or square), and **size** (small, medium, or large). Since there exist 2 x 3 x 3 = 18 possible objects, and only four observed objects (positive instances of the "concept" to be learned), we can infer that some 14 negative instances remain. If necessary, we could enumerate these negative instances, but in general this is not required. The left part of Figure 1 presents a minimal discrimination network for summarizing these observed and inferred instances. The **color** attribute occurs at the top of the network, since it is most useful in distinguishing positive instances from negatives. The four terminal nodes with +'s cover the observed positive instances, while the remaining terminal nodes lead to the unobserved negative instances.

In order to transform a discrimination network into a classification tree, only one operation is required – we remove all branches that lead only to negative instances. Applying this transformation to the network at the left of Figure 1, we arrive at the classification tree on the right of the same figure. This tree has only four terminal nodes, each corresponding to one of the + nodes in the original network. The resulting taxonomy specifies that there are two basic classes of objects – black and white. However, each of these is further subdivided, the black objects into small circles and medium squares, and the white objects into medium circles and large triangles. Due to the manner of its construction, no unobserved object will be covered by this classification tree. Thus, the tree *summarizes* the observed objects, but does not go beyond the data in the sense that it makes any predictions; we will return to this feature later in the paper.

AN OVERVIEW OF DISCON

We have implemented DISCON, a conceptual clustering system that incorporates the method described above.* The program is based on an earlier system we constructed for learning from examples, and required only minor modifications for the conceptual clustering domain. The major difference between DISCON and the earlier systems lies in the evaluation function used to direct search through the space of discrimination networks. At each point in the learning process, Hunt's system selected the attribute whose values matched the most positive instances, while Quinlan's method incorporated a more sophisticated information-based measure. Thus, both systems could be viewed as carrying out a best-first search without backup through the space of possible networks. Although these evaluation functions eliminated many undesirable networks from consideration, they were not guaranteed to find the simplest network (the one with the smallest number of nodes) that summarized the data. Disjunctive concepts were especially troublesome to this approach, and since these always occur in conceptual clustering tasks, we developed an alternate method for directing the search process.

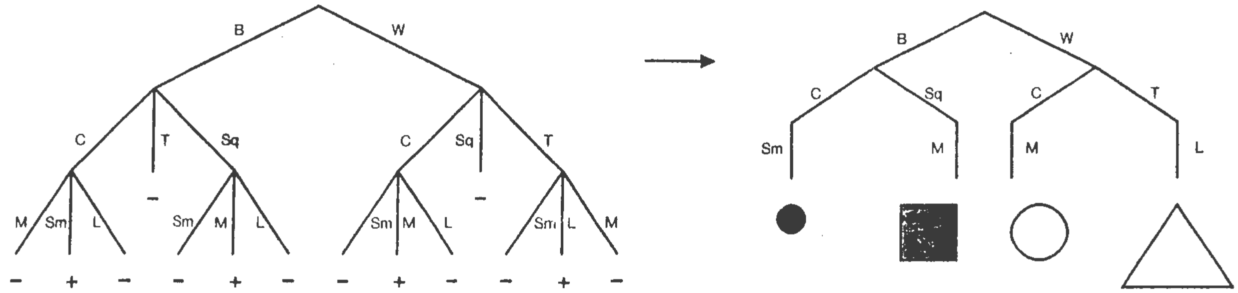


Figure 1. A discrimination network and a classification tree.

*DISCON stands for Discrimination-based Conceptual Clustering System. D. Fisher has independently proposed a very similar approach, and has implemented another conceptual clustering system that shares many features with DISCON [8].

Given a set of attributes, DISCON carries out a near-exhaustive look-ahead for each attribute, to determine the minimal subtrees possible for each value of an attribute. The numbers of nodes in these minimal subtrees are counted, giving a score for each attribute that corresponds to the evaluation functions used by Hunt and Quinlan. The lowest scoring attribute is selected for use at the top level in the classification tree, and one branch is created for each of its possible values, except for those having no associated observations. One could then apply this process recursively to determine lower parts of the tree. However, since DISCON determines the minimal subtrees during the look-ahead process, it takes advantage of this information, as we shall see directly.

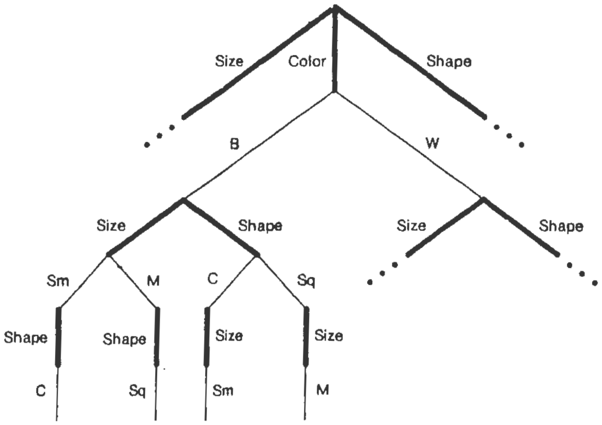


Figure 2. Searching the space of classification trees.

Figure 2 presents a partial search tree for the example considered earlier, involving four observations and the three attributes **color**, **size**, and **shape**. Note that two types of line occur in the figure. Bold lines represent branches in the search tree, while narrow lines represent branches in classification trees that are embedded within the search process. The reader should keep this distinction in mind as we describe DISCON's search method. The system begins by considering the attributes in their order of presentation; let us suppose that **color** is considered first. In order to determine the score for this attribute, DISCON begins constructing a classification tree with branches for the values **black** and **white**; these branches are shown with narrow lines. The program must then determine, for each of these branches, the size of the minimal tree that covers the observations. In order to do this, it must actually construct the subtrees; let us suppose that it deals with the **black** branch first.

Since the two attributes **size** and **shape** remain, DISCON must consider trees based on both choices. Let us suppose that it considers **size** first; this path is represented by the bold line labeled with this attribute. Now the system must construct a subtree based on the values of **size**; however, since only small black objects and medium black objects have been observed, only branches for **small** and **medium** are created. At this point, only the **shape** attribute remains, so this path in the search tree is taken in each case. Since only one observed object exists in these cases, only one new branch is necessary for each. Moreover, since these branches cover observed objects but none of the potential but unobserved objects, they lead to terminal nodes. Upon considering the subtree based on **shape**, the program notes that it is no more complex than the **size** subtree, and moves on to consider subtrees for the value **white**. After determining the minimal subtrees for both **black** and **white**, DISCON counts the number of nodes in these subtrees, giving a score of 11 for the **color** attribute. When the same process is applied to the **size** and **shape** attributes (at the top level), each receives a score of 12. Accordingly, DISCON selects **color** as the best attribute to use at the top of its classification tree.

Earlier we suggested that the same process could be applied recursively, letting DISCON determine the best attribute for the next level, and so on until terminal nodes were reached. However, note that the system must determine the minimal subtrees for the lower levels, during its computation of the scores for the highest level. DISCON takes advantage of this fact, and actually constructs each of the possible subtrees during its look-ahead process. However, the program connects the parent nodes only to the *minimal* subtrees, so that larger structures are effectively forgotten. As a result, many subtrees are constructed before the higher levels are created, but only a few of these are retained in the final taxonomy, which is the classification tree presented in Figure 1. Note that the system does not actually create a discrimination network, and then eliminate branches pointing to negative instances; rather, it never bothers to create these branches in the first place. Despite its apparent complexity, the DISCON program consists of some 75 lines of Franz Lisp code.

Table 2. Descriptions of four dinosaurs.

NAME	DIET	LEGS	DEFENSE	HIPS
BRONTOSAURUS	PLANTS	FOUR	SIZE	LIZARD
TYRANNOSAURUS	MEAT	TWO	TEETH	LIZARD
TRICERATOPS	PLANTS	FOUR	ARMOR	BIRD
TRACHODON	PLANTS	TWO	SPEED	BIRD

AN EXAMPLE: CLASSIFYING DINOSAURS

Now that we have examined DISCON's method for constructing classification trees, let us consider its behavior on a more realistic example. As we noted earlier, numerical clustering techniques have been used within biology to aid in the classification process, and one should be able to apply conceptual clustering methods here as well. We presented the system with descriptions of four dinosaurs, each having four associated features, as shown in Table 2. The features were **diet** (either **plants** or **meat**), number of **legs** (either **two** or **four**), method of defense (**size**, **armor**, **speed**, or **teeth**), and type of **hip-bones** (either **bird-like** or **lizard-like**). These descriptions are somewhat whimsical, since they are rather removed from direct observations of fossils, but they will serve for our purposes.

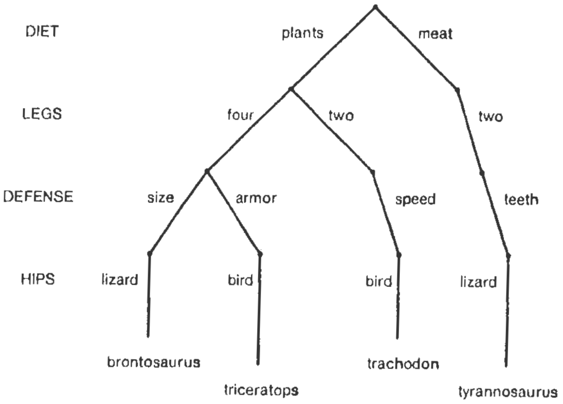


Figure 3. A taxonomic scheme for dinosaurs.

The method we have described is guaranteed to find a classification tree with the minimum number of nodes, but it is quite possible that more than one minimal tree exists. In such cases, the tree generated by the system is determined by the order in which the user has listed the attributes. Other things being equal, DISCON prefers attributes listed earlier to those given later. For instance, given the observations in Table 2 and the attribute order **number of legs**, **diet**, **method of defense**, and **type of hips**, the system generates the classification scheme shown in Figure 3. This divides the observed organisms into two groups based on their diet. Only one meat-eater is observed, with two legs, teeth for defense, and lizard-like hips. However, three plant-eaters exist, so this group is further sub-divided on the basis of number of legs. Only one biped exists, which uses speed for defense, and has bird-like hips. Since two quadrupeds are noted, these are divided into two further groups, one using size for defense and with lizard-like hips, and the other with armor for defense and bird-like hips. Given the descriptions in the table, this is a perfectly reasonable taxonomy, and some readers may have arrived at a very similar one themselves.

Figure 4 presents a quite different taxonomy, formulated by the system when the attributes were presented in the reverse order — **type of hips, method of defense**, diet, and **number of legs**. This scheme initially divides the organisms into two groups based on their type of hips, with two members of each class. The dinosaurs with bird-like hips are both plant-eaters, but they are further subdivided on the basis of number of legs. The quadruped uses armor for defense, while the biped employs speed to the same end. The dinosaurs with lizard-like hips are divided into two subclasses based on diet, with the plant-eater having four legs and using size for defense, and the meat-eater having two legs and using teeth for defense. This organizational scheme is very similar to that arrived at by paleontologists, who divide dinosaurs into two major groups (*Ornithischia* and *Saurischia*) based on their hip structures, and further divide these classes into groups similar to those shown in Figure 4. However, the generally accepted taxonomy is based on hundreds of organisms and many additional features, so the tree generated by our system should be taken with a grain of salt. Nevertheless, this example does reveal one potential application of our conceptual clustering method.

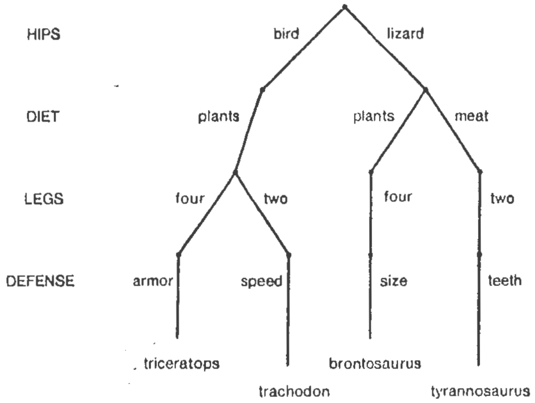


Figure 4. Another taxonomic scheme for dinosaurs.

EVALUATION OF THE APPROACH

In the preceding pages, we described a method for generating classification trees that relies on viewing this task as a variant on learning from examples. Like Michalski and Stepp's CLUSTER/2 program for taxonomy formation, the DISCON system also produces intensional descriptions of the classes it forms, and so provides another viable approach to the task of *conceptual* clustering. DISCON is based on Hunt's CLS and Quinlan's ID3 programs for generating discrimination networks, but does not require an explicit set of positive and negative instances, as did these earlier systems. Instead, the program treats observed objects as positive instances, and treats unobserved objects as negative instances, inferring the latter from known attributes and their values.

One difficulty with this approach is that DisCon's classification trees have no predictive power. Although they summarize the observed objects, they can never place a previously unobserved object in an appropriate class. A natural solution to this problem presents itself, which we plan to implement in future versions of the system. Rather than attempting to construct a network that completely discriminates positive instances from the inferred negative instances, DisCon should cease to make distinctions below a certain point. This cutoff might be stated in terms of the depth of the classification tree, or a more sophisticated test might be used. For example, Quinlan's [7] "change in information" measure could be employed to determine whether further branches will provide sufficient improvement in discriminating power to be worth including. If not, then no further distinctions need be made. This proposal also has a beneficial side effect, since it will keep DISCON from carrying out an exhaustive look-ahead. Instead, the system will need to look ahead only to the depth of the tree, and this will greatly improve the system's performance when many attributes are involved.

DISCON's approach to conceptual clustering should be applicable to any set of observations that can be stated in terms of attributes with symbolic values. One advantage of the CLUSTER/2 program over the current system is that the former can also handle observations involving numeric values and structured (or hierarchical) values. However, DISCON has two major advantages over Michalski and Stepp's system — its simplicity and its efficiency. Although we do not have statistics to back up our claim, we believe that our approach to conceptual clustering is inherently more efficient than the earlier one. We also believe that the existing system can be extended to deal with numeric and structured values without a substantial increase in either the program's complexity or in the search it must carry out. When DISCON has been extended in this manner, we will be able to run the system on the same clustering tasks reported by Michalski and Stepp [2], and compare the results of the two systems directly. However, independent of questions about elegance and efficiency, we now have two distinct approaches to the problem of conceptual clustering, and thus the potential for a better understanding of this fascinating discovery task.

One additional limitation of the current system should be corrected in future versions. As it stands, DISCON cannot deal with *relations* between objects. This ability could be included by providing the system with a list of n-ary predicates, such as greater-than. Given such a list, the program could enumerate all possible relational tests, and then determine which of these were the most useful. Some effort would be involved in extending the system along this dimension and those mentioned above, but taken together, they should lead to a simple yet robust tool for automating the process of conceptual clustering.

REFERENCES

1. Spath, H., *Cluster Analysis Algorithms*, John Wiley and Sons, 1980.

2. Michalski, R. S. and Stepp, R. E., "Learning from observation: Conceptual clustering," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, eds., Tioga Press, Palo Alto, CA, 1983.

3. Lebowitz, M., "Concept learning in a rich input domain," *Proceedings of the International Machine Learning Workshop*, 1983, pp. 177-182.

4. Wolff, J. G., "Grammar discovery as data compression," *Proceedings of the AISB/GI Conference on Artificial Intelligence*, 1978, pp. 375-379.

5. Langley, P., Zytkow J., Bradshaw, G. and Simon, H. A., "Mechanisms for qualitative and quantitative discovery," *Proceedings of the International Machine Learning Workshop*, 1983, pp. 121-132.

6. Hunt, E. B., Marin, J., and Stone, P. J., *Experiments in Induction*, Academic Press, New York, 1966.

7. Quinlan, R., "Learning efficient classification procedures and their application to chess end games," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, eds., Tioga Press, Palo Alto, CA, 1983.

8. Fisher, D., "A hierarchical conceptual clustering algorithm", Unpublished manuscript

SOME ISSUES IN TRAINING LEARNING SYSTEMS AND AN AUTONOMOUS DESIGN

David Coles and Larry Rendell

Department of Computing and Information Science
University of Guelph
Guelph, Ontario, N1G 2W1, CANADA

ABSTRACT

Training is an important consideration in the operation of learning systems; introduction of inappropriate training data may cause inefficient, possibly impotent behaviour.

This paper presents a general training method based on observation of current performance capability. The idea is to sample system performance and select training problems taxing search limits. Experiments demonstrate the utility of the approach in terms of time efficiency.

1. INTRODUCTION

Training methods used for inductive systems have recently taken on greater significance with the re-emergence of learning within AI.

Inductive systems may generally be decomposed into **performance** and **learning** functions. The performance element (PE) interacts directly with an external environment (e.g. problem instances requiring solution), while the learning counterpart (LE) must induce or refine a strategy for future action based on performance data. Of the several factors which influence learning element behaviour, one is the quality of training problems presented: higher quality data processed by the PE enable the LE to extract more meaningful information, yielding greater improvement in performance.

One approach to increasing information quality would be to present more complex training problems to the system. In practise, however, early performance ability is a limiting factor [4,5]. Generally then, learning systems require problems complex enough to present new information to the learning element, but still solvable within current performance element capabilities.

2. CONSIDERATIONS

Learning invariably incorporates some search activity. An effective learning system must limit search by using a strategy often referred to as a control structure (CS). If the PE of a system relies on such a structure for guidance, then the role of the corresponding LE is to modify the structure to improve subsequent performance[a]. An autonomous learning system requires no human intervention in improving its capabilities. This specifies an additional component: the **instance selector** (TRAINER) must independently generate problem instances to train the system effectively. Further, a useful trainer is general. These requirements may be stated more precisely:

1. To be general, a training method must be independent of the host system (i.e. its PE and associated CS). In effect, a 'black box' is required. This will be elaborated subsequently.
2. Effective training affords strong learning ability at moderate cost. Although a mechanized trainer provides full autonomy, it may not prove as effective as external *passive* means [1] (e.g. human training [4]). Further, costs (evaluated in time and space measures) which are excessive may undermine any benefits of a mechanized trainer.
3. We have stated that learning systems require problems complex enough to present new information to the learning element, but still solvable within current PE capabilities. Initial performance is typically poor, since (in early unsupervised learning), the control structure is primitive at best; beginning problems must be simple. However, as performance improves (due to CS refinements), more difficult problems are appropriate. In short, learning is a dynamic process, and requires an adaptive training mechanism. After estimating current PE performance ability, this mechanism determines an appropriate level of training problem difficulty.

Our training method requires two measures: a measure of **difficulty** for training problems, and a measure of

* This research was supported by an NSERC operating grant and an NSERC summer scholarship.

[a] This model is adapted from Buchanan et al [1].

performance for guaging current PE ability. Problem difficulty can be quantified as the length of the **shortest known** solution path from initial to goal states[b]. Performance may be measured as the number of nodes developed *(D)* in finding the goal state (fewer nodes developed means better performance). As cost considerations impose a practical limit on this search, define an upper bound on *D* as *D-MAX* (externally specified). *D-MAX* conveniently quantifies PE ability limits. We would expect that a system could have no more information presented to its LE than through a problem which requires exactly *D-MAX* states to be developed in reaching the goal; problems of greater difficulty must necessarily exceed *D-MAX*, resulting in no solution. To present a maximum of information to the learning element then, training problems should result in some *D* which is slightly less than *D-MAX*.

TRAINER estimates an appropriate level of difficulty for training problems, based on real time sampling of system performance. A suitable difficulty is one causing the performance element to develop slightly fewer than *D-MAX* nodes in reaching the goal state. This training scheme is not straightforward, since variance in performance may be large despite apparent similarity of sample data. Performance depends on the control structure (a more informed CS will perform better and more uniformly while an imperfect CS may be unable to deal with certain problem instances). While this large fluctuation could be compensated by increasing sample size, there are obvious limits imposed by cost.

3. TRAINER MODULE

Essentially TRAINER is a sampling mechanism which assesses the ability of a PE using any control structure. Its operation involves a movable window which iteratively approaches an upper bound on performance *(D-MAX)*. After a short series of window placements, a difficulty *(DIFF)* is found which approximates *D-MAX*. Figure 1 illustrates the operation of the TRAINER module.

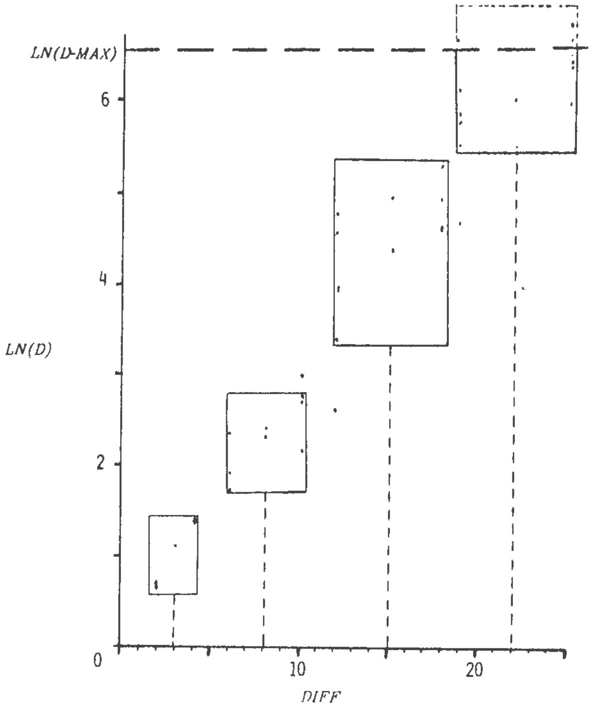


Figure 1. TRAINER OPERATION. Given a current CS, TRAINER samples PE ability, discovering performance *D* as a function of problem difficulty *DIFF*. The desired value of *DIFF* is that which results in *D* approximating *D-MAX*.

The module is currently applied to a data-driven learning system: PLS1 [5]. PLS1 provides a good testing environment, as it requires performance-sensitive training for strong learning behaviour (refer to next section).

To reiterate, difficulty *(DIFF)* and nodes developed *(D)* are necessary measures (and a value for *D-MAX* is assumed). TRAINER must select an appropriate set of depth *DIFF* problems which result in an approximation of *D-MAX* developed states. Our strategy is to approach *D-MAX* in terms of *DIFF* using a conservative underestimate. Regression is used for estimation of *D* as a function of *DIFF*. The general algorithm follows:

[b] Formally, this difficulty characterizes some region in instance space [4]. Our training mechanism requires some method of producing problems within this region. For some problem types, instances may be generated by making a number of random moves away from the goal state. However, other problems are governed by operators which are not uniquely reversible. Also, games typically contain a number of goal states. Instances in these categories may be generated by playing complete games or solving problems, retaining a list of nodes developed during search, and backing up a number of states to the desired difficulty. Training examples may be spontaneously generated using this method, or selected from a set of problems catalogued by difficulty (termed *active instance selection* in [2]).

1. Choose a range of difficulty values centered about a central value *(DIFF)* - initially, this value is arbitrarily small. Generate (or select) a number of problems at each edge of this 'window'[c], and refer these to the application system PE for solution, obtaining a measure of performance *(D)* for the control structure in each case[d].
2. Use a least-squares fit and a (log-)linear model to express *D* in terms of *DIFF*. Determine a value of *D* centered within the window. Take some (externally defined) fractional step between this central value and *D-MAX* , and translate into a new value for *DIFF* , using parameters obtained from the regression.
3. Repeat these steps until the neighbourhood of *D-MAX* is reached (e.g. within 90%). The final value of *DIFF* is the appropriate training difficulty for current PE ability. As the control structure improves, this difficulty level increases until the PE is capable of solving random depth problems.

The reason for using a progressive series of window placements is related to complexity and uncertainty. If too difficult a problem is attempted, no corresponding value of *D* can be found; this waste is expensive. Alternatively, for simple problems, inaccuracy of the linear model and significant variance in sample data may cause an inaccurate fit (step 2). Rather than extrapolating a final value of *DIFF* based on one sample of problems, it is ultimately more accurate and cost-effective to take several intermediate samples, so as not to exceed *D-MAX*; all windows except the last one or two approaching *D-MAX* require insignificant resources. The question of whether the TRAINER justifies its cost is examined in the following section.

4. RESULTS

To measure the effectiveness of the TRAINER, it was compared with a manual procedure for choosing problem difficulty using the fifteen-puzzle and the learning system PLS1 [5]. This is an iterative system so a **static difficulty vector** (SDV) was supplied. This vector defines a fixed progression of problem difficulty over successive iterations. The SDV used was one found to be effective in extensive human interaction with PLS1. Of several tested, it produced acceptable learning characteristics over the greatest proportion of runs. For the experiments, 36 trials were run for each of (i) TRAINER, and (ii) the SDV.

Table 1 summarizes the results:

	# Successful	**Expected Cost**
(i) USING SDV		
(504,000 nodes)		
within 75% of best	12 (33%)	42,000
within 25% of best	6 (16%)	84,000
(ii) USING TRAINER		
(870,000 nodes)		
within 75% of best	36 (100%)	24,000
within 25% of best	27 (75%)	32,000

Table 1. Within each of the two training environments, performance of resulting control structures is subdivided into two categories: within 75% and within 25% of performance using the best known CS. Column one indicates the number successful in each category (and corresponding percentage), while column two indicates the expected cost in nodes for a CS to demonstrate performance within the respective categories. Using the SDV, 504,000 nodes were developed over 36 runs. While use of TRAINER resulted in a greater number of developed nodes (870,000), the proportion of successes was dramatically higher, resulting in lower expected costs for each category.

Experiments with TRAINER to date demonstrate consistent and inexpensive behaviour. If within 25% of best is categorized as strong learning behaviour, the expected costs of achieving this using TRAINER is 32,000 nodes, a saving of approximately 68% over the case where an SDV is used. This is significant.

Yet the strongest support for use of TRAINER lies in its provision of complete autonomy for the host system. The effectiveness of external training may be attributed to extensive knowledge of system performance; such knowledge is expensive to acquire. By providing strong learning capabilities with only modest real time machine costs, TRAINER eliminates this other considerable expense.

[c] Draper and Smith [3] indicate that a sample distribution with exclusive emphasis on end points of a range results in the smallest variance in slope determination.

[d] The actual number of problems is determined by the statistical error of performance values sampled (95% confidence is used).

5. CONCLUSIONS

The intent of this method is to present a maximum of new information to any learning system, while abiding by its performance limitations. This is achieved by estimating performance results in terms of training problem difficulty. Training problems are autonomously generated according to results of performance sampling, and no assumptions are made about system capability or operation. Measures of system performance *(D)* and problem difficulty *(DIFF)*, and a limit on *D (D-MAX)* are the only requirements. Our method is effective and inexpensive, demonstrated by its current implementation, and it is general enough to have wide application within machine learning.

REFERENCES

1. B.G. Buchanan, C.R. Johnson, T.M. Mitchell, and R.G. Smith, "Models of Learning Systems", in J. Belzer (Ed.), Encyclopedia of Computer Science and Technology 11, Dekker, (1978), pp. 24-51.
2. P.R. Cohen and E.A. Feigenbaum,(Ed.), Handbook of Artificial Intelligence III, Kaufmann, (1982), pp. 360-372.
3. N.R. Draper and H. Smith, Applied Regression Analysis (2nd ed.), Wiley, (1980), 709 pp.
4. T.M. Mitchell, "Learning and Problem Solving", Proc. Eighth International Joint Conference on Artificial Intelligence, (1983), pp. 1139-1151.
5. L.A. Rendell, "A new Basis for State-space Learning Systems and a Successful Implementation" , Artificial Intelligence 20, (April 1983), pp. 369-392.

INDUCTIVE LEARNING OF PHONETIC RULES
FOR AUTOMATIC SPEECH RECOGNITION

Renato De Mori*, Michel Gilloux**
*Concordia University, Montreal, Canada
** Centre National d'Etudes des
Telecommunications, Lannion, France

ABSTRACT: An application of Automatic Inductive Learning of Rules from Examples to Automatic Speech Recognition os described. An algorithm for incremental learning is proposed and preliminary results are reported.

1. Motivations and Relations with Previous Words

A number of researches on Automatic Speech Recognition (ASR) have been carried out using a recognition model based on feature extraction and classification [DE MORI 83] and [BAHL 83]. With such an approach, the same set of features are extracted at fixed time intervals (typically every 10 msecs.) and classification is based on distances between feature patterns and prototypes [LEVINSON 81] or likelihoods computed from a markov model of a source of symbols generated by matching centisecond speech patterns and prototypes [BAHL 83].

These methods are usually speaker-dependent and are made speaker independent by clustering prototypes among many speakers. The classifier is not capable of making reliable decisions on phonemes or phonetic features, rather it may generate scored competing hypotheses that are combined together to form scored word and sentence candidates.

If the protocol exhibits enough redundancy it is likely that the cumulative score of the right candidate is remarkably higher than the scores of competing candidates.

If there is little redundancy in the protocols, like in the case of connected letters or digits or in the case of a lexicon of more than 10,000 words, then it is important that ambiguities at the phonetic level are solved before hypotheses are generated. For example, in the case of connected letters, in order to distinguish them between /p/ and /t/ the place of articulation is the only distinctive feature and its detection may require the execution of special sensory procedures on a limited portion of the signal with a time resolution finer than 10 msec.

This suggested to introduce plans for hypotheses generation and disambiguation [DE MORI 82].

Operators of these plans may translate a description of acoustic properties into more abstract descriptions or they may extract new useful properties. Operators may contain the execution of sensory procedures. Their application is conditioned by the verification of some preconditions in the database that contains already generated descriptions of the signal under analysis.

The planning system is a network of plans. The input to the network is made of descriptions of acoustic properties obtained by hybrid (parametric and syntactic) pattern recognition algorithms and the outputs are hypotheses about phonetic features. A portion of the network for the generation of hypotheses about the phonetic feature "nonsonorant-interrupted-consonant" (NI) is shown in Fig. 1. For more details on the features used in this example see [DEMICHELIS 83]. Each box in Fig. 1 represents an operator capable of producing descriptions of acoustic properties of phonetic hypotheses. Preconditions for operator application are logical expressions of predicates. Predicates are defined over relations between acoustic properties. Both precondition expressions and predicate definitions are not known a priori and have to be learned. Furthermore, the use of an operator in a plan is required only if that operator is the most suited for producing descriptions that are required for evaluating preconditions of operators that have already been considered for building that plan. So learning also involves building new sequences of operator applications when a feature is not correctly hypothesized. So far, most of the attention has been focused on learning and generating precondition relations. Learning of plans has mostly been done with the interaction of a human expert.

2. Learning Methodology

Learning rules from examples can be seen as the process of generalizing descriptions of positive and negative examples and previously learned rules to form new candidate rules. When applied incrementally this methodology can produce results which depend on the order in which examples are supplied and on the occurence of examples which are exceptions to the relevant rules. Incremental learning of rules has to come out with a set of rules that is the most consistent with the examples encountered so far.

In order to allow dynamic preservation of consistency among the set of rules, an algorithm is proposed which uses the Truth Maintenance System formalism [DOYLE 79] and which is reminiscent of previous work by Whitehill [WHITEHILL 80].

The choice of a description language for examples and rules along with that of the generalizing algorithms is critical in a learning system in the sense that it may or may not allow the learning of relevant rules.

A description language and rule generalization heuristics have been defined based on knowledge about rule-based Automatic Speech Recognition (ASR). This knowledge has been acquired with many years of experience. A relevant aspect of the learning system developped for ASR is that generalization rules are not constrained by the Maximally Common Generalization property introduced in [WHITEHILL 80].

Positive and negative facts used for learning operators preconditions are described by their relevant concept and a conjunction of predicate expressions. Each predicate expression or selector [MICHALSKI 83] asserts that an acoustic property has been detected or that an acoustic parameter has been extracted with some specified value.

A generalization rule derives from two conjunctions C1 and C2 a conjunction C3 that is more general than both C1 and C2, i.e. C1 ⇒ C3 and C2 ⇒ C3.

The generalized rules themselves are the nodes of a TMS [DOYLE 79]. Each node represents a rule of left-hand-side (LHS) CONJ and right-hand-side (RHS) CONC, having a support list SL whose IN and OUT part

[LEVINSON 81] S. Levinson, L.R. Rabiner, "Isolated and Connected Word Recognition Theory and Selected Applications", IEEE Trans. on Communications, Vol. COM-29, Num. 5, May 1981, pp. 621-269

[MICHALSKI 83] R.S. Michalski, "A Theory and Methodology of Inductive Learning", in Machine Learning: an Artificial Intelligence Approach, Tioga, 1983, pp. 83-134.

[WHITEHILL 80] S.B. Whitehill, "Self Correcting Generalization", Proc. of AAAI-80, 1980, pp. 240-242.

TABLE 1: the Learning Algorithm

```
procedure LearnExample (EXAMPLE)
    NEWNODE := NIL
    For every node N in ListOfNodes do
       if RHS(N) = Concept(EXAMPLE)
       then
          begin
             if MoreGeneralThan(LHS(N),Conjunction(EXAMPLE)
             then Push(EXAMPLE),PE(N));
             if Equivalent(LHS(N),Conjunction(EXAMPLE))
             then NEWNODE := N
       end
       else
          if MoreGeneralThan(LHS(N),Conjunction(EXAMPLE))
          Then Push(EXAMPLE,NE(N)) ;
    unless NN # NIL do
      begin
         NN := MakeNode(Conjunction(EXAMPLE),Concept(EXAMPLE))
         AddNode(NN)
      end
    if P(PE(NN),NE(NN)) and EveryIn(In(SL(NN))) and EveryOut(Out(SL(NN)))
    then TruthMaintain(NN,in) else TruthMaintain(NN,out)
endproc

procedure AddNode (NODE)
    for every node N in ListOfNodes do
       unless N = NODE do
         if RHS(N) = RHS(NODE)
         then
           case
             MoreGeneralThan(LHS(N),LHS(NODE))
               Push(NODE,In(SL(N)))
             MoreGeneralThan(LHS(NODE),LHS(N))
               begin
                 Push(N,In(SL(NODE)))
                 PE(NODE) := Union(PE(NODE),PE(N))
               end
           else
             begin
               NC := Generalize(LHS(NODE),LHS(N)),RHS(NODE)))
               AddNode(MakeNode(NC,RHS(NODE)))
             end
         else
           case
             MoreGeneralThan(LHS(N),LHS(NODE))
               Push(NODE,Out(SL(N)))
             MoreGeneralThan(LHS(NODE),LHS(N))
               begin
                 Push(N,Out(SL(NODE)))
                 NE(NODE) := Union(NE(NODE),PE(N))
               end

endproc
```

are respectively the list of nodes with RHS CONC and LHS less general than CONJ and the list of nodes with RHS different of CONC and LHS less general than CONC. With each node are kept the lists of consistent examples (PE for positive evidence) and unconsistent examples (NE for negative evidence). Lastly each node as a STATUS property which is IN when the correspondence rule is believed to be true and OUT otherwise. A node is IN i.e. its STATUS is IN if and only if all the nodes in the IN part and all the nodes in the OUT part of its SL are respectively IN and OUT and the numbers of examples in PE and NE satisfy a given predicate P (for example NE $\gtrsim$ 2.PE).

When a new example is learned a new node is created if necessary and this node is generalized with the existing ones to generate new nodes that are themselves generalized with other ones. Then the PE and NE of concerned nodes are updated and STATUS properties are modified when necessary and propagated through the network in order to maintain consistency. The stability of this process is guaranteed together by the definitions of SL and the predicate P. The algorithm is described in Table 1 in a Pascal-like form.

For each concept encountered so far a characteristic rule is derived from the network of nodes whose LHS is the disjunction of LHSs of all IN nodes with corresponding RHS.

3. Results and Conclusion

With this method, precondition rules for ten operators have been learned from the prounounciation of 700 Canadian postal codes leading to a very accurate generation of hypotheses about phonetic features like NI.

The following is an example of the precondition of the operator that generates the hypotheses Ni:

([AC2=MediumPeak] [F3-F1=[700,750]] [Burst=Detected]
[Buzz=NotDetected] [Step=NotDetected])
([AC1=ShortDeepDip] [AC2=LongPeak]
[A2=A1=[-0.5,0.0]] [Burst=NotDetected]
[Step=NotDetected]) ([AC1=LongDeepDip]
[AC2=LongPeak] [F2=[1200,1300]] [F3-F2=[1600,1700]]
[F3Locus-F3vowel=[200,250]] [Burst=Detected]
[Buzz=NotDetected]) [Step=Detected]

AC1, AC2 are morphology descriptions of the signal energy, Step is a predicate on the morphology description of the signal envelope sampled every 2.5 msecs. Fi are formant frequencies, Ai are formants amplitudes extracted by the "prevocalic transient analysis operator". Burst and Buzz are predicates defined over envelope and buzz descriptions.

ACKNOWLEDGEMENT

This work was supported by Natural Sciences and Engineering Research Council of Canada, Grant No. A2439.

REFERENCES

[BAHL 83] L. Bahl, F. Jelinek, R.L. Mercier, "A Maximum Likelihood Approach to Continuous Speech Recognition", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, Num. 2, March 1983.

[DEMICHELIS 83] P. Demichelis, R. De Mori, P. Laface, and M. O.Kane, "Computer Recognition of Plosive Sounds Using Contextual Information", IEEE Transactions on Acoustic Speech and Signal Processing, Vol. ASSP-31, Num. 2, April 1983, pp. 359-377.

[DE MORI 82] R. De Mori, A. Giordana, P. Laface, L. Saitta, "An Expert System for Interpreting Speech Patterns", Proc. of AAAI-82, 1982, pp. 107-110.

[DE MORI 83] R. De Mori, Computer Models of Speech Using Fuzzy Algorithms, Plenum Press, N.Y., 1983.

[DOYLE 79] J. Doyle, "A Truth Maintenance System", Artificial Intelligence, Vol. 12, Num. 3, 1979, pp. 231-272.

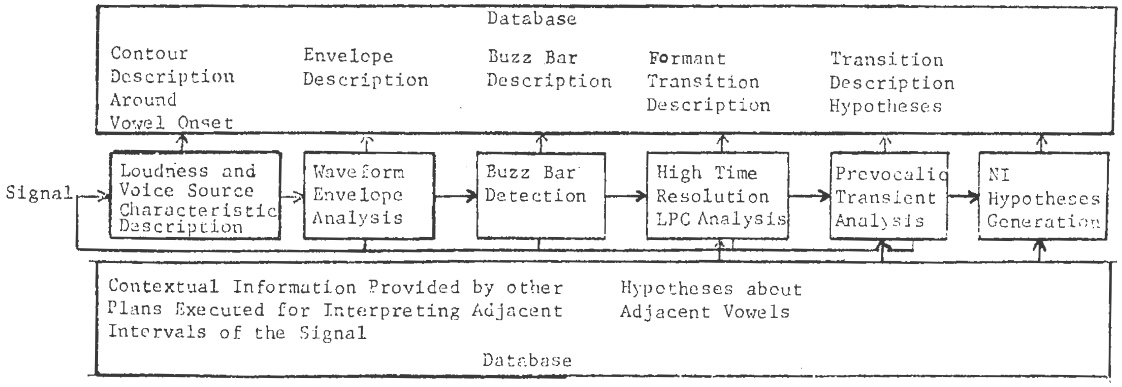

Database
Contour Description Around Vowel Onset
Envelope Description
Buzz Bar Description
Formant Transition Description
Transition Description Hypotheses
Signal
Loudness and Voice Source Characteristic Description
Waveform Envelope Analysis
Buzz Bar Detection
High Time Resolution LPC Analysis
Prevocalic Transient Analysis
NI Hypotheses Generation
Contextual Information Provided by other Plans Executed for Interpreting Adjacent Intervals of the Signal
Hypotheses about Adjacent Vowels
Database

Fig. 1: A Plan for Generating Hypotheses about the Phonetic Feature NI.

APPLYING TEMPORAL CONSTRAINTS TO THE PROBLEM OF STEREOPSIS OF TIME-VARYING IMAGERY

Michael Jenkin

Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4

Abstract

A noncooperative algorithm is presented for the problem of the stereopsis of time-varying imagery. The algorithm integrates two problems; the problem of stereopsis and the problem of tracking objects through time. Rather than finding the intersection of the two problems to be more difficult, it was found that by solving the two problems simultaneously, and thus incorporating the spatio-temporal context within which a scene exists, some of the hard subproblems belonging to the problems of stereopsis and temporal correspondence could be avoided.

In terms of the model for temporal stereopsis presented in this paper the induced effect and hysteresis have simple explanations as a result of the motion of the stretching stimuli. It is argued that the difficulty algorithms for static stereopsis have with the induced effect and hysteresis are a result of their temporal nature.

The algorithm relies on a general smoothness assumption to assign both disparity and temporal matches. A simple model of the motion of three-dimensional points is used to guide the matching process and to identify conditional matches which violate the general smoothness assumption. A proximal rule is used to further restrict possible matches.

The algorithm has been tested on both synthetic and real input sequences. Input sequences were chosen from three-dimensional moving light displays and from "real" grey-level digitized images.

Introduction

A basic problem in any motion understanding vision system is that of determining a temporal correspondence between objects. Independent of the domain of application, be it human body motion [11], Hamburg street scenes [1], or human hearts [15] in order to analyze the motion of the objects it is first necessary to determine the position of objects in the scene as a function of time. In a perspective projection of the world such a correspondence can be difficult to obtain unless some assumption is made as to either the type of object being observed (such as being rigid or being composed of jointed rigid parts[16]) or the type of motion the object can exhibit [17]. The problem arises because in a perspective view true object location is not readily available. The human visual system overcomes this problem in part by using two images of the scene rather than just one. By combining the two views depth information can be obtained and thus accurate positional information is available. Such information would simplify the problem of determining temporal correspondence.

Any algorithm for stereopsis has to face two critical design choices; what monocular point descriptions are to be used, and what mechanisms are to be used to resolve any ambiguity within the population of possible local matches. When the stereopsis problem is expanded to include time-varying imagery, the algorithm must also deal with the problem of tracking the monocular point descriptions, or the three dimensional descriptions which they represent through time. In addition a choice must be made as to the order of processing. Temporal matching could be performed before, after, or simultaneously with stereopsis.

If stereopsis is performed after temporal matching then temporal matching must take place in a two dimensional perspective view of the world. Without some motion or object assumptions rotation in depth, and plastic deformation would present difficult problems to such a scheme. Alternatively, if stereopsis takes place before temporal matching the temporal matching is easier as it can be performed in real world co-ordinates. Unfortunately the stereopsis problem would then reduce to that of solving static stereoscopic images. From the work of Marr and Poggio [9] and Mayhew and Frisby [10] we know that even when sophisticated monocular primitives such as zero-crossings or zero-crossings and peaks are available the problem of static stereopsis is very difficult, and usually cannot be solved without some global mechanism to assign final disparities. Performing stereopsis simultaneously with temporal matching seems best as the temporal matching can be performed in real world co-ordinates (thus avoiding the problem of perspective matching) and motion information could be used to further limit the possible binocular matchings.

An algorithm using this approach has been previously described[6]. This approach made certain simplifying assumptions about the nature of the input, which would be difficult to overcome in order to apply the algorithm to real image data. The algorithm assumed that there was no occlusion, that all points were visible in all frames, and that all real world positions were known at some initial time. We base our algorithm on the same underlying principles as Jenkin [6] but present one more suitable to the analysis of real world images. In particular we must deal with the problem of occlusion; a feature visible by one eye may not be visible by the other, and points may move out of view of the entire vision system.

Psychological Basis

We based our algorithm on the assumption that the motion of an object being viewed would obey a general smoothness assumption.

The smoothness assumption was based upon certain beliefs from psychology. The principle of least action [14] suggests that when we perceive an object as moving we tend to perceive it as moving along a path that in some sense is the shortest, simplest or most direct. Ramachandran and Anstis' principle of "visual inertia"[13] suggests that when any object moves in one direction at uniform velocity we tend to perceive it as continuing its motion in that direction. These rules formed the basis of our smoothness assumption

The temporal stereopsis algorithm follows the basic smoothness assumption; changes in feature characteristics will be small and continuous. In particular we assume that when observing the motion of an object (one that does not disappear or is created), the object will be relatively unchanged from one frame to the next:

1) The location of a given feature would be relatively unchanged from one frame to the next.

2) The three-dimensional vector velocity of a given feature would be relatively unchanged from one frame to the next.

3) The motion of objects has temporal continuity, ie. the above two assumptions hold for sequential time intervals.

The first two assumptions define how moving objects (those that are not created or destroyed) act locally in time, while the third smoothness assumption says that the first two will hold over temporally sequential inputs, and allow decisions to be made in time.

Many algorithms for static stereopsis exist (see Marr and Poggio [9] for a review). Marr and Poggio [9] identify two physical constraints in order for the left and right view to be combined:

R1 Uniqueness. Each item from each image can be assigned at most one disparity value.

R2 Continuity. Disparity varies smoothly almost everywhere.

Unfortunately R1 flatly contradicts the usual interpretation of Panum's Limiting Case (ie the interpretation that a feature in one eye's image can be matched to more than one feature in the other eye's image). Mayhew and Frisby [10] have developed a better rule of uniqueness: that a given point in space may hold at most one feature at any given time.

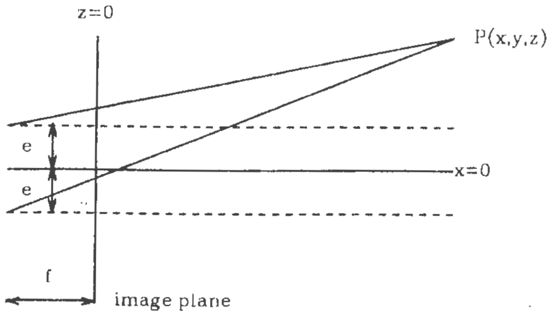


Figure 1: Nonconvergent Vision System.

Marr and Poggio [9] then use their R2 to allow for a "pulling effect" to make final disparity assignments. As they point out there is really no evidence to support such "pulling" or global labelling. Consider the standard "wedding cake" random dot stereogram (see Grimson [4] for an example). When viewed stereoscopically one sees several distinct layers with sharp boundaries. If spatial global constraints are used to assign disparity values using R2 above then one would expect the algorithm to produce smooth boundaries between the layers due to edge effects. Indeed this is an effect which is observed when algorithms using such approaches are applied to this type of stereogram. This contradiction is a strong argument against stereo algorithms using similar spatial global constraints.

The Algorithm

Figure 1 shows the basic geometry of the nonconvergent binocular vision system. The two eyes of the system are separated by a distance of $2e$ and are located at positions $(-e, 0, f)$ and $(e, 0, f)$, looking towards infinity in the minus z direction. (In Figure 1 the y axis can be considered to project out of the plane of the paper towards the reader.) By similar triangles it is easy to show that a point $P = (x, y, z)$ on an object will have a projection on the plane $z = 0$ (the image plane) in the right and left eye's co-ordinate systems respectively as:

$$x_r = \frac{(x - e) * f}{f - z} \qquad (1a)$$

$$y_r = \frac{y * f}{f - z} \qquad (1b)$$

$$x_l = \frac{(x + e) * f}{f - z} \qquad (2a)$$

$$y_l = \frac{y * f}{f - z} \qquad (2b)$$

Where f is the focal length, and (x_l, y_l) and (x_r, y_r) are the projections of the point P in the left and right eye views respectively.

Provided that the point P is visible by both the left and right eyes, then the three-dimensional co-ordinate of P can be reconstructed from these projections as:

$$x = \frac{e * (x_r + x_l)}{x_l - x_r} \qquad (3a)$$

$$y = \frac{2 * e * y_r}{x_l - x_r} \qquad (3b)$$

$$z = f - \frac{2 * e * f}{x_l - x_r} \qquad (3c)$$

by solving for x, y, and z in equations (1) and (2).

Let L_j and R_k be the co-ordinates of point j in the left eye, k in the right. The valid three-dimensional point which can be constructed from L_j and R_k using equation (3) will be denoted as $[j, k]$.

All possible combinations of L_j and R_k would result in roughly n^2 possible points from n real points in three-space. The number of matchings can be reduced by noting certain geometric properties of the nonconvergent binocular vision system.

From equations (1) and (2) for a combination $[l, r]$ to be valid it must be true that the y co-ordinate of l in the left eye (denoted as $y(L_l)$) must be equal to the y co-ordinate of r in the right eye (denoted as $y(R_r)$). Thus from the geometry $[l, r]$ will be valid only if $y(L_l) = y(R_r)$. For real image data, this constraint should be relaxed.

In addition, from equations (1a) and (2a) and as e is positive, and as z must be negative, then for $[l, r]$ to be a valid combination $x(L_l) > x(R_r)$. As the image plane is not infinite in size as depicted in Figure 1, there will be a minimum distance a point can lie from the image plane before it becomes visible to both eyes. In the limiting case $z([l, r]) = 0$ is the closest a point may lie to the eyes of the system. In this case $x(L_l) - x(R_r) < 2e$ from equation (3).

Let τ_y the maximum amount of deviation between $y(R_r)$ and $y(L_l)$, and let τ_x be the restriction on $x(R_r)$ and $x(L_l)$ resulting from the finite size of the left and right eye [illegible] using the following two constraints.

$$| y (l_l) - y (R_r) | < \tau_y \qquad (4a)$$

$$0 < x (l_l) - x (R_r) < \tau_x \leq 2e \qquad (4b)$$

Figure 2 shows how the parameters τ_x and τ_y restrict the region for possible spatial matchings. Let the circle represent some feature in the left eye's image. If we superimpose the left and right images then the rectangular region formed by τ_x and τ_y denotes the region in the right eye's view from within which we may look for possible matches.

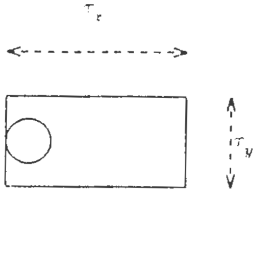


Figure 2: Spatial Constraints on Matchings.

In relationship with the human visual system, the nonconvergent binocular system is a limiting case in which the eyes are fixed on a point an infinite distance from the viewer. The constraints τ_x and τ_y denote the region of fusion for the binocular system and correspond to Panum's fusional area in the human visual system. Results from Psychology indicate that although matches must be limited to this area, it is possible to have more than one match per feature from within this area.

The problem can now be stated more formally: given $P_1(t)$ to $P_n(t)$, the three-dimensional location of the features at time t, $V_1(t)$ to $V_n(t)$, the vector velocities of the same n points at time t, and L_1 to L_n, and R_1 to R_n (where L_i does not necessarily correspond to R_i), the left and right eye views of the points at the next time interval (denoted by $t+1$), the problem is to find mappings $\varphi_l : N \to N$ and $\varphi_r : N \to N$, where $N = \{1,\dots,n\}$, such that P_i has location $[\varphi_l(i), \varphi_r(i)]$ at time $t+1$. Note that P_i may have 0, 1 or more possible corresponding points at time $t+1$. In addition, the algorithm must construct a list of points that have been "created" or that appear at time $t+1$. Formally, it must find the points that are elements of
$\{[l, r] \mid l \in L \wedge r \in R \wedge \sim\exists P_i : [l, r] = [\varphi_l(i), \varphi_r(i)]\}$,
where $L = \{L_1, \cdots, L_n\}$, $R = \{R_1, \cdots, R_n\}$, and $[l, r] = [\varphi_l(i), \varphi_r(i)]$ if the two points correspond to the same three-dimensional location.

Certain results from Psychology suggest that the number of possible matches in φ_l and φ_r can be restricted by considering the velocity and distance parameters of the points. From work on apparent motion [7] it is clear that spatial range limits the power of attraction; "when the distance between the circles is increased the quality of motion is sometimes affected, so that maintaining the perception of a smoothly moving figure may require some change of intervals or of the flash duration." Korte's third law of apparent motion [8] states that the increase in optimum difference in time is essentially proportional to any increase in separation in space, or equivalently, that the critical time increases linearly with distance. There is also the "visual momentum" effect reported by Ramachandran[13]. Thus the velocity of the object is also a limiting factor in object tracking. These two constraints can be incorporated within a general smoothness assumption by defining predicates *velocity* and *distance* which limit membership in φ_l and φ_r as:

$$distance(i, j, k) = | P_i(t) - [j, k] | < \tau_d \qquad (5a)$$

$$velocity(i, j, k) = \left| V_i(t) - \frac{([j, k] - P_i(t))}{isi} \right| < \tau_v \qquad (5b)$$

where isi is the inter-stimulus time interval, the notation $|a|$ denotes the length of the vector a, and τ_d and τ_v are constraints limiting the distance an object can move, and the change in velocity an object can exhibit during a given frame. Characteristics of monocular primitives that are used to restrict combinations of $[l, r]$ could also be used to restrict φ_l and φ_r.

Consider the restriction of the tracking problem to two dimensions. Figure 3 shows how the values of τ_d and τ_v restrict the possible motions of an object being tracked. The small circle moving from left to right represents the motion of the feature being tracked. Suppose we are tracking it from its second position and that it has a constant velocity. The circle with radius τ_d shows the region which constrains the object's next position using equation (5a). The object has been restricted from moving more than a distance τ_d between frames.

The circle with radius τ_v restricts the region in which the object may move based on the maximum change in velocity an object may undergo between frames. This is the geometrical interpretation of equation (5b). The circle is centred at the predicted location of the feature in the next frame and has radius τ_v. As both the *distance* and *velocity* predicates must hold we need look for valid matches only in the intersection of these two regions.

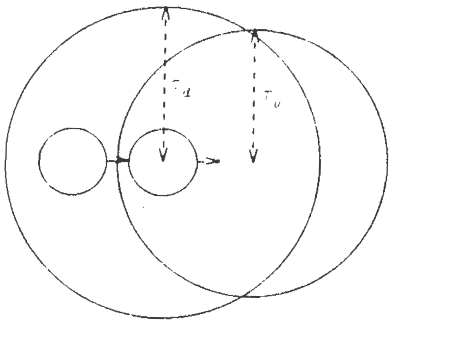


Figure 3: Motion Constraints on Matching.

In three dimensions the circles become spheres and valid matches no longer are restricted to the plane. The intersection of the two spheres (which correspond to the two circles mentioned above) defines the region for possible matches.

The values of τ_d and τ_v must be set so that the motion of the object you wish to track does not violate *distance* and *velocity*. In the limiting case the two spheres must at least overlap so that it is possible to satisfy the conditions for a valid match.

The smoothness assumption permits sharp turns, and even for an object to change its direction (see Figure 3). There is a trade-off between change in direction and change in scalar speed. An object may radically change its direction but only by reducing its speed. Similarly an object may change its speed, but only by moving in a similar direction.

τ_d and τ_v define mathematically the conditions of the smoothness assumption for a single interstimulus interval. An object will be said to satisfy the general smoothness assumption for a given frame if for that frame it satisfies the predicates *distance* and *velocity*.

Unless the situation is ideal, or the values of τ_x, τ_y, τ_d, and τ_v are chosen so as to unreasonably constrain the problem there will still be a large number of possible combinations $[l,r]$ and possible mappings φ_l and φ_r. In static stereopsis, even with sophisticated monocular primitives such as zero-crossings [9] or zero-crossings and peaks [10], not all local matches will have been reduced. A common approach has been to use a global spatial relaxation process (in a possibly limited manner) to assign final disparity values. As it has been shown earlier, this is not an attractive approach.

One possible approach would be to apply some sort of relaxation on the vector velocity values of the features. Such an approach would have a psychological basis in the rule of common fate. There would be some difficulties with non-rigid objects and with scenes containing more than one object moving with different velocities. This would, however, be an interesting starting point for future research.

Rather than use any sort of global mechanism to determine the final spatial and temporal correspondences, decisions are made locally in the spatial dimension but are made later in the temporal dimension. A very simple model for the motion of three-dimensional features was constructed. By considering possible temporal combinations of motion hypotheses inconsistencies can be identified. These inconsistencies can be used to locally trim hypotheses to determine the correct spatial and temporal matches.

For the purposes of this discussion let circles represent features. One of the following labels (see Figure 4) can be identified with the between frame motion for each point:

CREATE
: The point is created from the void.

DEATH
: The point disappears.

TRACK
: The point moves from one frame to the next

SPLIT
: The point undergoes fission to become two or more points in the next frame.

In addition two or more points in one frame may have the same candidate point for tracking in the next. Thus branches of SPLIT and TRACK labellings may also be identified as MERGE hypothesis.

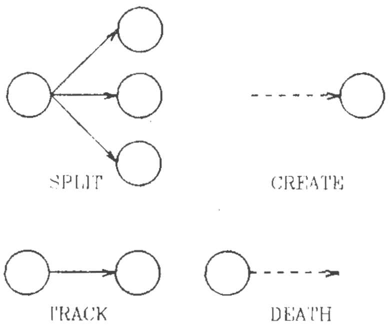


Figure 4: Motion hypotheses

In a world with perfect monocular feature detectors the set of tracking hypotheses would be much simpler. As no perfect monocular feature detecter exists it is quite possible that SPLIT and MERGE hypotheses are not just the result of poor matching, but that the monocular features do SPLIT and MERGE. Take a sheet of reflective material and bend it back and forth, the specular reflection will move along the surface of the material, splitting and fusing. A monocular primitive that is sensitive to peaks in the intensity of the image would find that the monocular primitives do SPLIT and MERGE. It would be incorrect to remove SPLIT and MERGE hypothesis in a real world application simply because most real world objects do not undergo these effects.

Possible labellings are assigned to a point $P_i(t)$ as follows. Let $M_i = \{[l,r] \mid l \in L \wedge r \in R \wedge$ *distance* (i,l,r) [illegible] $\}$. Then $P_i(t)$ is identified with a TRACK labelling if $|M_i| = 1$, it is identified with a SPLIT labelling if $|M_i| > 1$ and it is identified with a DEATH labelling if $|M_i| = 0$. Points $[l,r]$ which are not associated with any M_i are considered for the CREATE label [illegible]

$|M_i|$ counts the number of possible motion hypotheses for a given point P_i. If $|M_i| = 1$ then there is only one motion hypothesis for the point P_i, and it can be associated with a TRACK labelling. If $|M_i| > 1$ then the point P_i has competing motion hypotheses. At this point it is not possible to determine if the point actually splits into two or more points, or if some of the possible motion hypotheses are invalid. If $|M_i| = 0$ then the point P_i has no possible motion hypotheses. This point is said to have died. Finally, some combinations $[l,r]$ will not have been candidates for any point P_i. These points are considered for creation hypotheses.

Let $P_i(t)$ be a possibly empty set of points for which both motion and positional information is known. Examine the monocular primitives for the next frame and construct all possible matches $[l,r]$. From $P_i(t)$, $V_i(t)$, and $[l,r]$ all possible motion hypotheses according to the model given above can be constructed. Rather than deciding the correct matches at this point, assume (temporarily) that all of the matches are correct. Thus each possible match $P_i(t) \to [l,r]$ is considered to give rise to a new point at time $t+1$. The process of examining the monocular input is repeated for time $t+2$ and all labellings are then considered. Thus there are double mappings of the form $P_i(t) \to [l,r] \to [l',r']$.

The algorithm assumes that every possible matching $P_i(t) \to [l,r]$ is correct. This assumption is the prediction make by the prediction phase of the algorithm. For each possible match $P_i(t) \to [l,r]$ a point (with corresponding positional and velocity parameters) is constructed. The constructed points are temporarily assumed to be valid at time $t+1$. Possible matchings are then considered between the constructed point at time $t+1$ and all points constructable from the monocular features presented to the algorithm at time $t+2$. Inconsistencies between the mapping $P_i(t) \to [l,r]$ and $[l,r] \to [l',r']$ are used by the test phase of the algorithm to restrict the double mappings and to decide on the correct mapping $P_i(t) \to [l,r]$.

There are two problems within the general problem of stereopsis of time-varying imagery. The first is that there may be ghosts arising from the problem of static stereopsis, and the second is that while tracking points TRACK type results are preferable to SPLIT. Consider the tracking problem alone for a moment; assume that the motion of the features obey a general smoothness assumption, then the points will move smoothly from one frame to the next. A problem will arise only when for a given frame there exists more than one candidate point for tracking. If the incorrect choices arise due to noise (or from some other random source such as the accidental alignment of true features in an image), then one would expect that although they satisfy the smoothness assumption for the current frame, they will not satisfy it for a sequence of frames. A large class of these types of errors could be eliminated by simply examining the "matching potential" of the points for the next frame. If the points are truly invalid then they should die out as there will be no matches for these points in future frames which do satisfy the smoothness assumption. When discontinuities are present in the motion of the features (such that the motion violates the general smoothness assumption), the points will die. Once the discontinuity ends the points will be recreated and tracked correctly.

In order to trim the possible matches it seems that these are two cases of interest; when a CREATE label is followed by a DEATH label, and when one branch of a SPLIT labelling is followed by a DEATH label. The hypothesis reduction is summarized below (see table 1).

This hypotheses trimming can be demonstrated graphically (see figures 5, 6, and 7). There are two entries in table 0.1 which are inconsistent with the general smoothness assumption. These can be used to eliminate possible matchings (treat them as GHOST matchings). The first is when a CREATE labelling is followed by a DEATH labelling. Such an occurrence is depicted in Figure 5. Some accidental combination of monocular features has given rise to a valid three-dimensional point in one frame which then disappears in the next. This "creation" of an object is considered to be a GHOST labelling and it is removed from the list of valid hypotheses.

Current	Next			
	DEATH	**CREATE**	**TRACK**	**SPLIT**
DEATH	-	-	-	-
CREATE	**GHOST**	-	**CREATE**	**CREATE**
TRACK	**TRACK**	-	**TRACK**	**TRACK**
SPLIT	**GHOST**	-	**SPLIT**	**SPLIT**

Table 1: Hypothesis reduction.

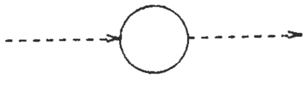


Figure 5: CREATE followed by DEATH.

The second inconsistency occurs when a branch of a SPLIT labelling is followed by a DEATH labelling. Such an occurrence is displayed in Figure 6. One branch of the three way SPLIT is shown to have a DEATH labelling in the following frame while all other branches have TRACK labellings. The set of hypotheses is trimmed by considering the branch of the SPLIT that was followed by a DEATH labelling as a GHOST, and removing it from the list of hypotheses. In performing this operation the number of branches of the given SPLIT labelling has been reduced by one.

There is another possibility when trimming branches of a SPLIT labelling (see Figure 7). In this case all but one of the branches of a SPLIT labelling are followed by DEATH labels. When these branches are removed there remains a SPLIT label with only one branch. Consider the value of M_i. It will have been reduced from the condition $|M_i| > 1$ to $|M_i| = 1$. The SPLIT labelling of this node is replaced by TRACK to reflect the new value of $|M_i|$.

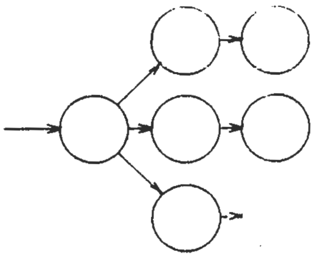


Figure 6: SPLIT followed by DEATH.

This trimming follows from the general smoothness assumption. When a label is identified as a GHOST it is removed from the list of possible labels. Thus it is possible that in reducing a SPLIT label that there exists a different branch of the SPLIT which has been reduced due to trimming to a TRACK. This is a very desirable result.

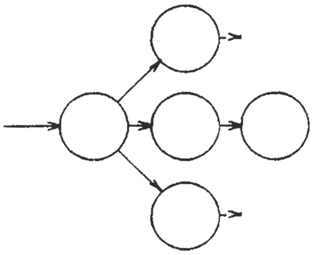


Figure 7: Reducing SPLIT to TRACK.

At this stage the final correspondences must be determined. All surviving labels have been found to satisfy the general smoothness assumption. For all, except SPLIT labellings, final tracking and disparity assignments have now been obtained. For SPLIT labellings a final restriction can be made to further reduce SPLIT labels to TRACK, or to reducc the number of branches in the SPLIT labelling.

As mentioned earlier, distance limits the effect of attraction for motion correspondence. This effect is most apparent when a SPLIT labelling is being considered. Kolers [7] reported two experiments which showed that in a SPLIT situation the presence of near points reduced the attraction of far ones. This can be used as a final constraint to remove some competing SPLIT branches. Remove competing SPLIT branches that lie at a distance greater than some fixed multiple k of the distance that the minimum distance SPLIT branch lies from the father point in the previous frame. This has the added affect of favouring no motion hypotheses for stationary points having a stationary candidate point for tracking.

The use of k to remove branches of SPLIT labelling will not reduce all SPLIT labels to TRACK. Setting the value of k to 1 will reduce most SPLIT labels, but any point which splits into two equidistant points will not be reduced to a TRACK labelling. It is not clear that all SPLIT labelling, should be reduced, as some inputs may exhibit objects which undergo fission, and SPLIT labels must be retained if these points are to be correctly tracked.

The proximity rule used to obtain final matches is dependent upon the motion of the candidate feature with the smallest possible displacement. Trimming is performed in terms of multiples of the shortest distance. As this takes into account inputs with different speeds, or different inter-stimulus intervals, the choice of k will not be dependent upon, either the velocity of the features, or the length of inter-stimulus interval.

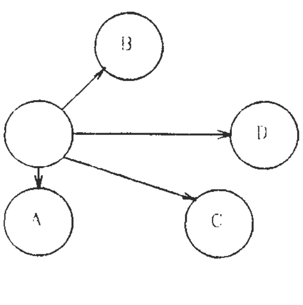


Figure 8: Final Decisions using k .

Figure 8 shows how the value of k is used to remove SPLIT branches. Figure 8 shows a point in one frame with arrows showing valid SPLIT branches labelled A through D in the next frame. Suppose that the value of k was 2. Then as all branches that lie at a distance greater than k times the distance to the nearest branch are eliminated, the nodes labelled B and D would be trimmed.

Each of the modules interact in a cyclic fashion. The results of the decision module become the basis of the prediction module for the next frame of processing. Thus this is a small closed system which observes the motion of the features and builds up an observer oriented $2\frac{1}{2}$-D sketch of the motion of the objects over time. The sketch is temporally limited in that it holds information only for two or (indirectly through motion information) three frames.

When all of the final decisions have been made from time t to time $t+1$, the points that were tracked to, or created at time $t+1$ become the points for which tracking is required. The values of P_i and V_i are updated, with created points given an initial velocity of zero.

The algorithm does not weight against MERGE hypotheses except that inconsistencies are used to reduce the total number of possible matchings, and thus indirectly the MERGE hypotheses are removed. There appear to be two choices; either treat a MERGE as simply being a mapping to more than one point having the same physical location, or actually merge the points to form only one single point.

Why is this a problem? Consider the case of two points at the same depth moving at constant velocity at right angles having a frame in which there exists a single point of intersection. Depending upon the representation of a MERGE, different effects will be perceived by the algorithm. If more than one point is allowed to occupy the same physical location at the same time then there is no problem as each point will be seen to continue in a straight line. If points are forced to merge then the velocity of the merged point must be computed. It is not obvious how this should be done, and depending on the choice different effects will be perceived.

Neither approach seems to be particularly attractive. Aside from the physical impossibility of having more than one point occupy the same physical location at the same time there are other problems in allowing MERGEs to be treated as simple TRACK hypothesis. Unless the motion is actually a TRACK and not a MERGE there will be two points for tracking that will be indistinguishable by the algorithm: they will have identical velocity and motion parameters. This would not be desirable. The other choice is unattractive for reasons that have already been mentioned. What is really desired is an approach that has the advantages of both schemes, but with none of their problems. The problem is this; if while tracking points are merged together, (assuming some function to obtain the velocity of the new point), what process should be used in the following frame to SPLIT the point apart if the merge was really a collision? If more than one point is allowed to occupy the same physical location how are points truely merged together once a true merge has been detected?

This problem was solved by allowing MERGEd points to be multiple points in that more than one point may have the same physical location provided they have different velocity parameters. Whenever two (or more) points have identical positions in the six-space $(P_x, P_y, P_z, V_x, V_y, V_z)$ a true MERGE was assumed to have taken place (as the points can not be distinguished by the system). Otherwise the points are considered to be distinct. This has the advantage that it solves both problems without having to remember 'special' points that may (or may not have) been merged together.

Experimental Results

Figures 9, 10, and 11 show the action of the algorithm on a three-dimensional moving light display of two colliding hexagons. The moving lights outline two hexagons, a stationary one (the one on the left), and one moving at constant velocity from left to right. In order to help a viewer distinguish between the two hexagons the moving hexagon has an additional marker located at its centre. Figure 9 shows the search window determined from *distance* and *velocity* applied to one frame of the input. The values of τ_d and τ_v were taken to be equal, and the previous frame's motion was marked with white lines. The darker grey regions delineate the regions in which one of *distance* and *velocity* hold, with the brighter regions representing their intersection. Note that for the stationary points the two regions coincide.

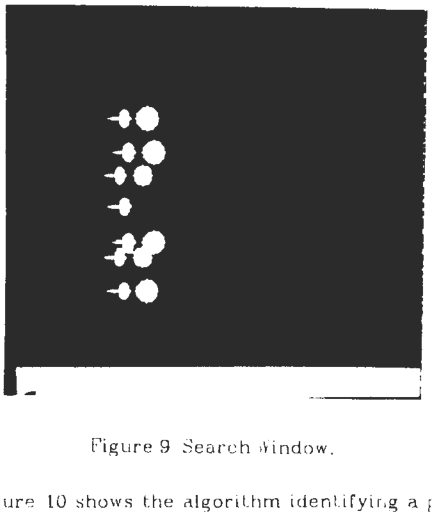


Figure 9 Search Window.

Figure 10 shows the algorithm identifying a potential TRACK labelling. As each potential labelling is identified the display is updated. Note that a previous TRACK labelling has already been identified for a different pair of points (indicated by the white line). Figure 11 shows the action of the trim phase of the algorithm as applied to a stereoscopic ghost formed from the accidental allignment of the two hexagons. The stereoscopic ghost was eliminated as the creation of the ghost and it's subsequent movement violated the smoothness assumption.

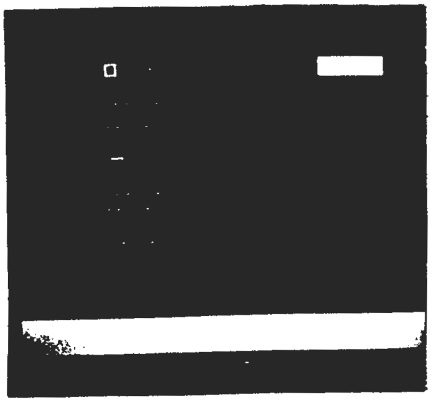


Figure 10: Identifying Potential Labels.

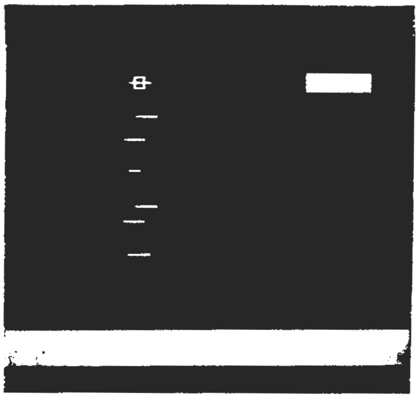


Figure 11: Removing Stereoscopic Ghosts.

The algorithm has also been applied to real world data. A digitized stereoscopic film was created of a small toy boat. The Moravec operator [11] was applied to each frame, and the points detected by the operator were used as input to the tracking algorithm (see Figure 12). Figure 13 shows the tracking of the feature points identified by the moravec operator. The tracked points were transformed to the same perspective view as the image of the boat in Figure 12, and then superimposed on the image.

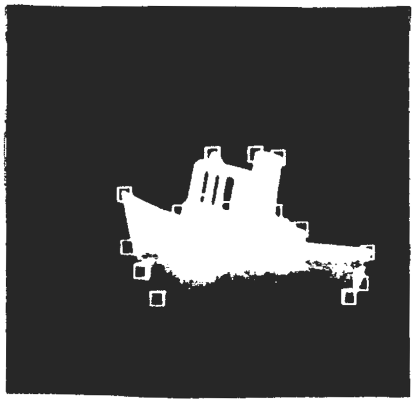


Figure 12: Moravec Response.

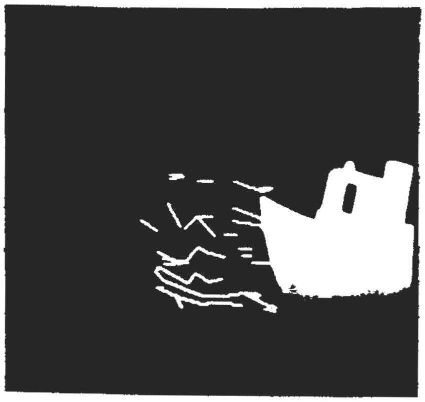


Figure 13: Tracking Grey-level Images.

Discussion

We have presented an algorithm for the stereopsis of time varying imagery. The algorithm was based on a general smoothness assumption; a feature point will remain relatively unchanged from one frame to the next. The algorithm is conservative in that provided that the features being satisfy the general smoothness assumption and a proximity constraint then the correct matches will be included in the tracking observed by the algorithm. Competeing matches are eliminated if they (i) violate the general smoothness assumption over he next two frames, or (ii) violate the proximity rule.

The algorithm utilizes spatio-temporal context constraints, and takes a wait and see approach to both the temporal matching and stereopsis problems. The algorithm performs the short range matching function, in that if a given object leaves the system view and then returns the two objects will not be identified as being the same. It is assumed that there exists some long range matching function to interpret the results of the temporal stereopsis algorithm.

Hysteresis is the effect that Panum's fusional area can be extended once fusion has been obtained. Algorithms for static stereopsis have used this as a basis for global mechanisms in stereopsis. A simpler interpretation (in terms of the temporal stereopsis algorithm) is that the value of τ_c differs for creation and motion hypotheses In particular for the motion hypothesis there is a broader horizontal range allowed for $[l,r]$. In terms of a general smoothness principle such a favouring of a motion hypothesis is justified in that the temporal continuity of an object is more desireable than having to destroy the object and then re-create it.

A similar intepretation exists for the induced effect of vertical disparities. The value of τ_y can be relaxed for motion hypothesis. Again in terms of a general smoothness constraint such an approach is justified.

In order to modify the temporal stereopsis algorithm so that it would exhibit expansion of the fusional area once fusion had been obtained, it was necessary to introduce two new parameters: e_x and e_y into the algorithm. e_x and e_y denote the fusional region once fusion has been obtained. As the fusional region should expand for motion hypotheses, $e_x \geq \tau_x$ and $e_y \geq \tau_y$, where τ_x and τ_y denote the fusional region for creation type hypotheses (see Figure 14).

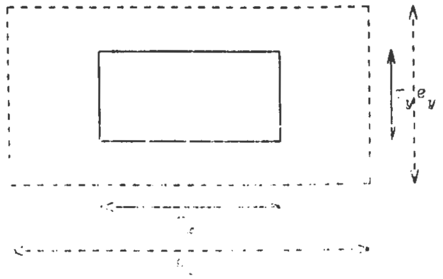


Figure 14: Creation and Motion Fusion Areas

Conclusions

We have developed a spatially noncooperative algorithm for the stereopsis and tracking of time varying stereoscopic imagery. We have argued that by enforcing temporal consistencies based on results from human psychology both stereoscopic ghosts and invalid temporal matches can be eliminated. Our algorithm correctly interprets Panum's Limiting Case in that we allow for more than one match for a given monocular feature, and we have suggested that the induced effect and hysteresis are not effects of static stereopsis but rather result from the relaxation of spatial constraints in order to maintain temporal stereopsis.

References

[1] Dreschler, L., and Nagel, H.-H., On the Frame-to-Frame Correspondence between Greyvalue Characteristics, Proceedings of the 7th IJCAI-81, Vancouver.

[2] Duda, Richard O. and Hart, Peter E., Pattern Classification and Scene Analysis, John Wiley and Sons, New York, 1973.

[3] Fender, D., and Julesz, B., Extension of Panum's fusional area in binocularly stabilized vision, J. Opt. Soc. Am., 57, 1967.

[4] Crimson, W. E., From Images to Surfaces, MIT Press, 1981.

[5] Hoffman, D. D. and Flichbaugh B. E., The Interpretation of Biological Motion, Massachusetts Institute of Technology A.I. Memo No. 608, 1980.

[6] Jenkin, M. R. M., Tracking Three Dimensional Moving Light Displays, Proceedings of the SIGGRAPH SIGART Interdisciplinary Workshop on Motion, Toronto 1983.

[7] Kolers. P. A., Aspects of Motion Perception, Pergamon Press Ltd., Oxford, 1972.

[8] Korte, A., Kinematoskopische Untersuchungen. Zeitschrift fur Psychologie, 72, 193-296, 1915.

[9] Marr, D., and Poggio, T., A theory of human stereo vision, Proc. R. Soc. London, B 204 (1979).

[10] Mayhew, J. E. W., and Frisby, J. P., Psychophysical and computational studies towards a theory of human stereopsis, Artificial Intelligence 17, 1981.

[11] Moravec, H. P., Robot Rover Visual Navigation, UMI Research Press, 1981.

[12] O'Rourke, J., Image Analysis of Human Motion, PhD Thesis. University of Pennsylvania, 1980.

[13] Ramachandran, V.S., and Anstis, S. M., Extrapolation of Motion Path in Human Visual Perception, Vision Res, Vol. 23, 1983.

[14] Shepard, Roger N., and Cooper, Lynn A., Mental Images and their Transformations, MIT Press, Cambridge, 1982.

[15] Tsotsos, J. K., A framework for Visual Motion Understanding, University of Toronto Technical Report CSRG-114, 1980.

[16] Ullman S., The Interpretation of Visual Motion, MIT Press, Cambridge, 1979.

[17] Webb, J. A., Structure from Motion of Rigid and Jointed Objects, Proceedings of the 7th IJCAI-81, Vancouver

Acknowledgements

I would like to thank John Tsotsos and Allan Jepson for their helpful suggestions and criticisms during the preparation of this paper.

This research was supported in part by the University of Toronto, and by the Defence Research Establishment Atlantic.

SCALE-BASED DESCRIPTIONS OF PLANAR CURVES

Alan K. Mackworth and Farzin Mokhtarian
Department of Computer Science
University of British Columbia

Abstract

The problem posed in this paper is the description of planar curves at varying levels of detail. Five necessary conditions are imposed on any candidate solution method. Two candidate methods are rejected. A new method that uses well-known Gaussian smoothing techniques but applies them in a path-based coordinate system is described. By smoothing with respect to a path-length parameter the difficulties of other methods are overcome. An example shows how the method extracts the major features of a curve, at varying levels of detail, based on segmentation at zeroes of the curvature, κ. The method satisfies the five necessary criteria.

1. The Problem: Detail and Scale

Achieving a proper notion of detail in a domain is a prerequisite for the construction of useful descriptions of domain elements. Such descriptions allow, for example, efficient coarse-to-fine matching. In vision, the problem of image detail is often reduced to the problem of scale. One approach to that problem extracts the locations of zero-crossings in the second derivative of the Gaussian-smoothed signal, varying the width of the Gaussian kernel to obtain multiple descriptions of the signal. This method has been used to extract "edge" elements at different spatial frequencies in an image intensity function I(x,y) of two independent variables [1]. It has also been used to perform automatic peak selection in histograms [2] and generalized to extract a new description, the scale space image, of signals that are functions of one variable [3].

Here, we are concerned with the problem of detail at a higher level in the visual system for the description of edges and other contours rather than for the extraction of edge locations from sensory data. We pose the problem of scale-based descriptions of planar curves. In our cooperative interpretation project [4], we are faced with the task of, for example, matching shorelines, roads and rivers extracted from aerial and satellite imagery, at varying scales, and from sketched maps. To do this successfully for a shoreline, say, we want to extract scale-based descriptions of it as an alternating sequence of headlands and bays.

2. Necessary Conditions on Any Method

In artificial intelligence, we often settle for sufficiency conditions, simply finding a method that will do the job. A more powerful methodology specifies criteria that ***any*** adequate solution method must satisfy. Here, we propose five such criteria.

Criterion 1 The method must be computational, preferably using local support techniques.

Criterion 2 The method must produce essentially the same result regardless of the coordinate system imposed on the curve. This implies that the descriptions must be well-behaved under rotation, translation, reflection and uniform expansion of the coordinate system (or the curve itself).

Criterion 3 It must not be ill-conditioned. Small changes in the curve should not cause large changes in its descriptions.

Criterion 4 The descriptions should correspond to human performance on the task.

Criterion 5 The method must not require arbitrary choices that affect the descriptions.

These criteria may not all be easy to justify or trivial to verify; however, if accepted, they impose stringent requirements on the class of all acceptable methods.

Our first candidate method was based on the detail hierarchy for curves used in the Mapsee project since its origin [5,6]. That hierarchy is a binary tree of straight line approximations to the curve. The initial approximation joins the end points. Subsequent approximations recursively refine the initial approximation by breaking the approximation at the point on the curve farthest from the straight line joining its end points.

This method violates criterion 3. If two points are roughly equidistant outliers from the current approximation then a small movement of one of them could cause a large change in the description. Criterion 5 is also violated. If the curve is closed then a purely arbitrary choice of end points is required which has a drastic effect on the description.

A second candidate method considers the curve to be a function of one variable y(x). If that function is multivalued, break it into several piecewise single-valued functions. Then apply Witkin's techniques [3] to y(x) to extract smoothed functions and mark the points of inflection. The problems with that method would include the handling of the boundary conditions at the end of each break in the curve. Even if those serious problems were solved the method would still not satisfy criterion 2. For example, after a reflection, of the coordinate system (or the curve) through the line $y = x$ the method would not produce essentially the same result. Smoothing x(y) with respect to y is quite different from smoothing y(x) with respect to x. Similar arguments apply to rotation transformations.

These considerations suggest using a description based on curvature [7,8] but one that is elaborated to analyze the curve at varying levels of detail in scale space.

3. A Method

To satisfy criterion 2, we have to use a path-based coordinate system for a curve C. Consider the parameterization

$$C = \{(x(t), y(t)) \mid t \in [0,1]\}$$

In this section we consider only closed curves so

$$x(0) = x(1) \quad \text{and} \quad y(0) = y(1).$$

The parameter t is a linear function of s, the path length along the curve from (x(0),y(0)), scaled to range over [0,1]. x(t) and y(t) may be considered defined over (-∞,∞) with periodic behaviour:

$$x(t+1) = x(t) \quad \text{and} \quad y(t+1) = y(t)$$

The method requires smoothing the functions x(t) and y(t) by convolution with a Gaussian kernel of width σ.

$$\text{Define } X(t,\sigma) = x(t) \circledast g(t,\sigma) = \int_{-\infty}^{\infty} x(u) \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(t-u)^2}{2\sigma^2}} du$$

and define $Y(t,\sigma)$ similarly.

The smoothed curve C_σ is simply:

$$C_\sigma = \{(X(t,\sigma), Y(t,\sigma)) \mid t \in [0,1]\}.$$

Notice that $C = C_0 = \lim_{\sigma \to 0} C_\sigma$.

The points of particular interest on C_σ are the points of inflection. The curvature κ of a planar curve at a point on the curve is the inverse of the radius of curvature of an osculating circle tangent at that point with its sign indicating the direction of curvature. Define

$$y' = \frac{dy}{dx} \qquad y'' = \frac{d^2y}{dx^2}$$

Then

$$\kappa = \frac{y''}{(1+(y')^2)^{3/2}}$$

The zeroes of κ are of interest. Since to obtain C_σ we are smoothing x(t) and y(t), it is cumbersome to return to the image domain to compute $y'(x)$ and $y''(x)$, in order to compute κ; moreover, there would be difficulties when $y' \to \infty$ or $y'' \to \infty$ and when y(x) is multivalued. Accordingly we wish to express κ purely as a function of derivatives of x(t) and y(t). Those derivatives can then be computed directly using appropriate masks. Define

$$\dot{x} = \frac{dx}{dt} \qquad \ddot{x} = \frac{d^2x}{dt^2}$$

$$\dot{y} = \frac{dy}{dt} \qquad \ddot{y} = \frac{d^2y}{dt^2}$$

Then

$$y'(x) = \frac{\dot{y}}{\dot{x}}$$

$$y''(x) = \frac{dy'}{dx} = \frac{\frac{d}{dt}\left(\frac{\dot{y}}{\dot{x}}\right)}{\dot{x}} = \frac{\dot{x}\,\ddot{y} - \ddot{x}\,\dot{y}}{\dot{x}^3}$$

and

$$\kappa = \frac{\dot{x}\,\ddot{y} - \dot{y}\,\ddot{x}}{(\dot{x}^2+\dot{y}^2)^{3/2}}$$

In the smoothed curve $\dot{X}(t,\sigma) = \frac{\partial X(t,\sigma)}{\partial t}$, $\dot{Y}(t,\sigma)$, $\ddot{X}(t,\sigma)$ and $\ddot{Y}(t,\sigma)$ are needed to compute $\kappa(t,\sigma)$. They can be obtained directly from x(t) and y(t) using,

$$\dot{X}(t,\sigma) = \frac{\partial X(t,\sigma)}{\partial t} = \frac{\partial[x(t) \circledast g(t,\sigma)]}{\partial t} = x(t) \circledast \left(\frac{\partial g(t,\sigma)}{\partial t}\right)$$

and

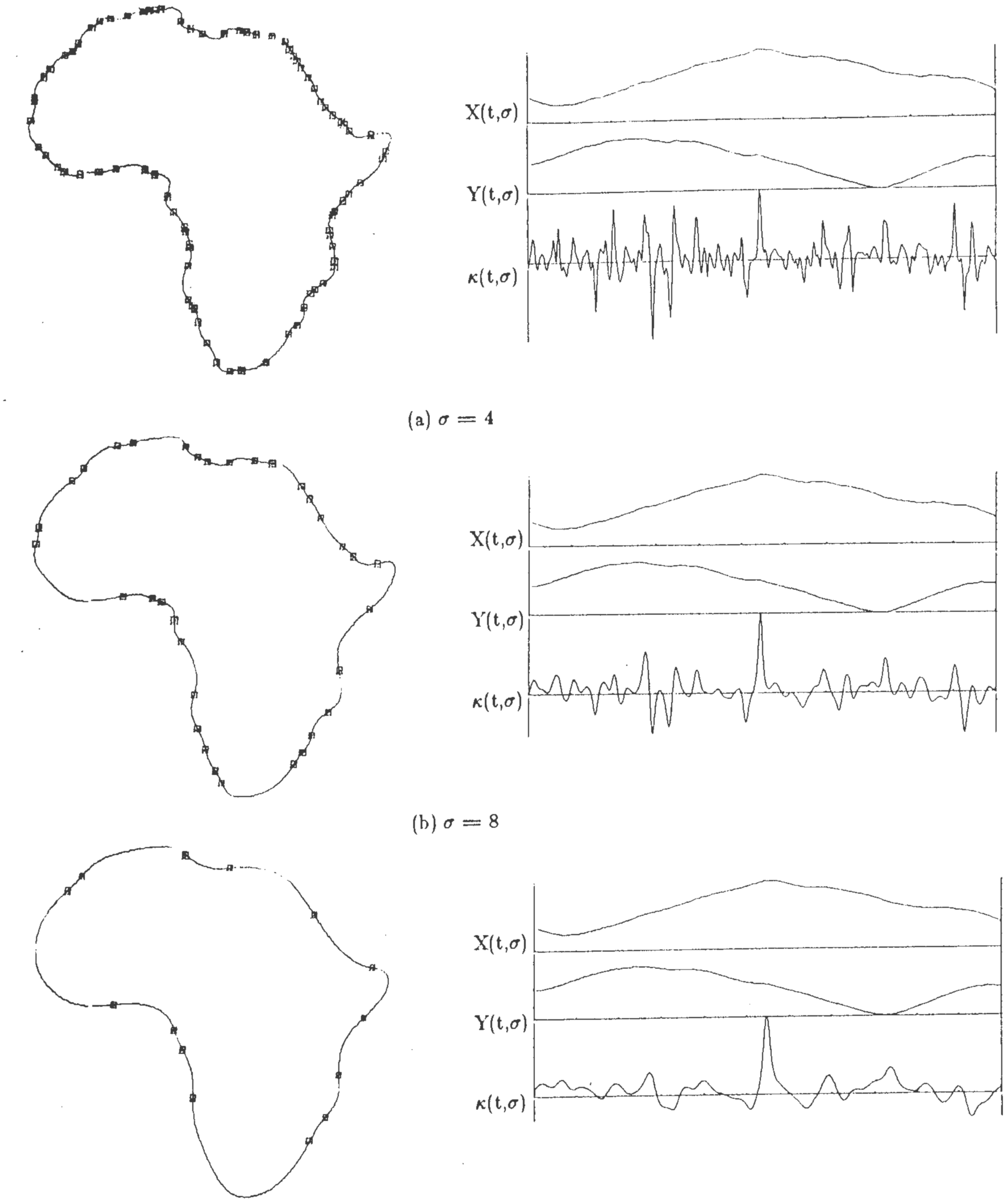
$$\ddot{X}(t,\sigma) = \frac{\partial^2 X}{\partial t^2} = x(t) \circledast \left(\frac{\partial^2 g(t,\sigma)}{\partial t^2}\right)$$

and similarly for $Y(t,\sigma)$. Using these equations $\kappa(t,\sigma)$ may be computed directly from convolutions performed on x(t) and y(t).

Several remarks about the method are appropriate. First, notice that for closed curves, treating x(t) and y(t) as periodic eliminates all edge effects. Second, if C is closed then the choice of the point on C at which $t = 0$ is purely arbitrary but has no effect on the description in terms of zeroes of κ or the smoothed curve. Third, the use of a path-based parameterization of the curve gives the desired invariance with respect to rotation, translation, reflection and uniform scale change of the curve. The curvature κ is invariant under rotation, translation and reflection. If the curve is scaled by a factor w then $\kappa' = \frac{1}{w}\kappa$. In particular the shape of the smoothed curve and the relative locations of the zeroes of κ will be invariant. One way to see this is to realize that linear coordinate transforms commute with linear smoothing operations. Fourth, small changes in the original curve may perturb the zero-crossing description for small σ but for larger values of σ their effects will disappear.

4. An Example

This method is applied to the coastline of Africa in Figure 1 for successively doubled values of σ. Beside each C_σ the functions $X(t,\sigma)$, $Y(t,\sigma)$ and $\kappa(t,\sigma)$ are displayed. The domain of t, the interval [0,1], has been divided into 1024 equally-sized subintervals for this experiment. The values of σ are given in terms of the number of subintervals. The locations at which $\kappa = 0$ are marked on each curve. As $\sigma \to \infty$ the curve asymptotically approaches its centre of mass. Notice also that as σ becomes larger the major headlands and bays emerge as dominant. At this point we can only appeal to the reader's intuitions to justify the claim that the results correspond to human performance.

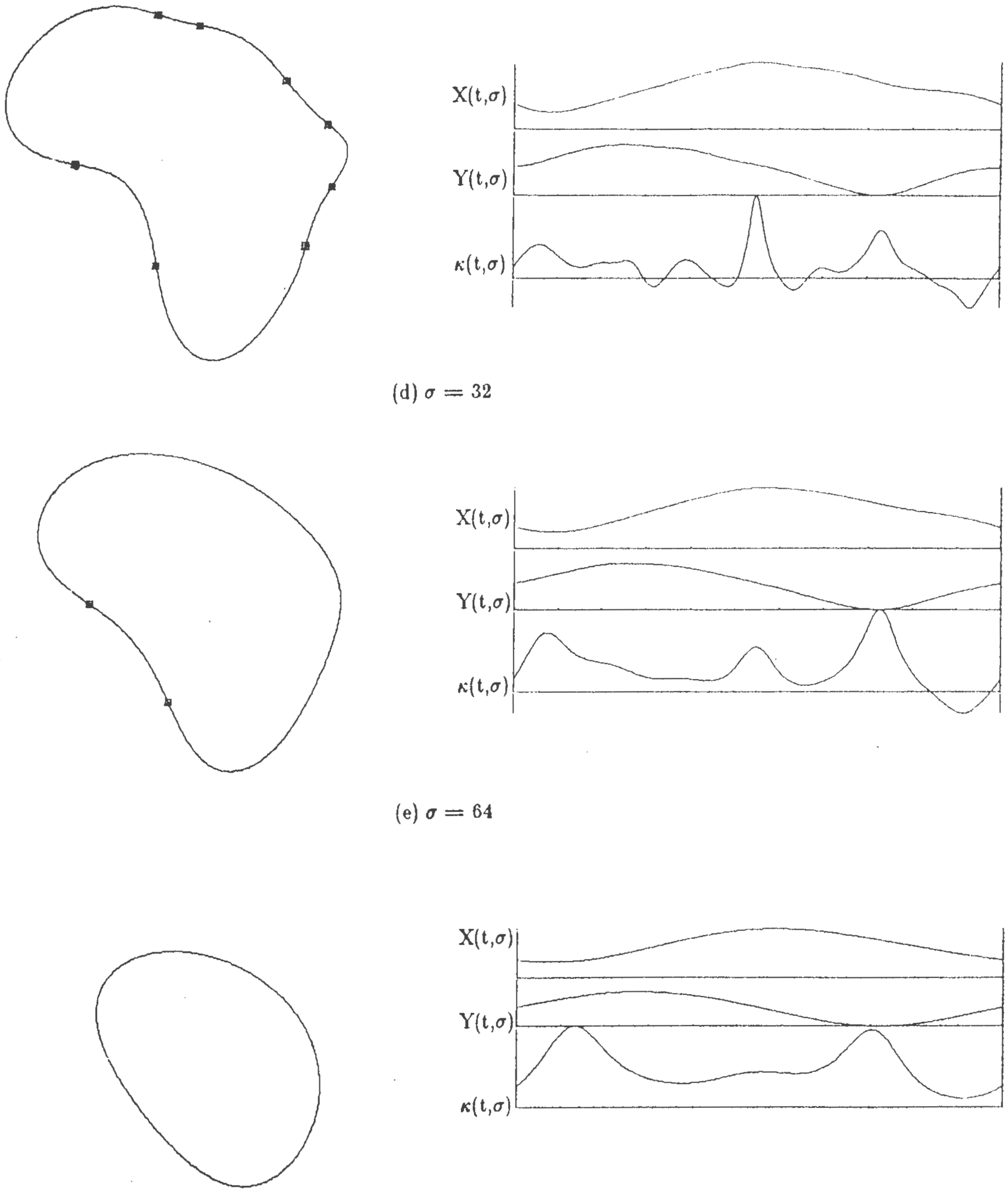


(a) $\sigma = 4$

(b) $\sigma = 8$

(c) $\sigma = 16$

Figure 1. Smoothing a Curve: Scale-based Effects



(d) $\sigma = 32$

(e) $\sigma = 64$

(f) $\sigma = 128$

Figure 1. (Continued) Smoothing a Curve: Scale-based Effects

5. Extensions

We have only discussed the application to closed curves so far. However, in our application, curves do not always close. They may also have free ends and junctions, or they may extend beyond the bounds of the map or satellite image - the frame problem. The only difficulty in extending our method to these curves lies in specifying the correct boundary conditions. Extensions to space curves and surfaces in higher dimensionality spaces should be pursued.

6. Conclusion

We have posed a problem of scale-based description of planar curves, proposed five criteria to judge any solution method and described a method that satisfies those criteria.

7. Acknowledgements

We are grateful to Jim Little for his comments on a draft of this paper. This research is supported by NSERC Operating Grant A0281.

8. References

[1] Hildreth, E.C. (1980) "Implementation of a Theory of Edge Detection", AI-TR-579, MIT Artificial Intelligence Laboratory.

[2] Glicksman, J. (1982) "A Cooperative Scheme for Image Understanding Using Multiple Sources of Information", Ph.D. Thesis, Dept. of Computer Science, Univ. of British Columbia.

[3] Witkin, A.P. (1983) "Scale-space Filtering", *Proc. Eighth Int'l Joint Conf. Artificial Intelligence,* pp. 1019-1022.

[4] Havens, W.S. and Mackworth, A.K. (1983) "Representing Knowledge of the Visual World", *IEEE Computer, 16,* 10, 90-96.

[5] Mackworth, A.K. (1977) "On Reading Sketch Maps", *Proc. Fifth Int'l Joint Conf. Artificial Intelligence,* pp. 598-606.

[6] Ballard, D.H. and Brown, C.M. (1982) *Computer Vision,* Prentice Hall.

[7] Hoffman, D.D. and Richards, W.A. (1982) "Representing Smooth Plane Curves for Recognition: Implications for Figure-Ground Reversal", *Proc. Am. Ass'n for Artificial Intelligence,* pp. 5-8.

[8] Duda, R. and Hart, P. (1973) *Pattern Classification and Scene Analysis,* Wiley.

IMPLEMENTING PROGRAPH IN PROLOG: AN OVERVIEW OF THE INTERPRETER AND GRAPHICAL INTERFACE

P.T. Cox; T. Pietrzykowski
Acadia University
Wolfville, Nova Scotia
B0P 1X0

Abstract -- The graphical programming language PROGRAPH and visual effects accompanying execution of prographs are briefly described. A correspondance is demonstrated between prographs and executable Prolog programs written in micro-PROLOG. It is shown how the graphical information can be expressed, and how the Prolog program corresponding to a prograph can be modified to incorporate communication with this graphical structure. Finally, an interpreter is described. This interpeter, written in micro-PROLOG, executes the modified Prolog program using the graphical information to produce the required visual effects. It also allows the user to query partial results during execution.

1. Introduction.

The recent strong interest in functional programming languages is due to disenchantment with procedural programming languages [1,4], and rapidly growing interest in dataflow architectures [7].

Functional programming languages themselves also have some drawbacks. For example, to avoid using variables one must either apply strong restrictions as in Lisp, where the functional nature allows only one return value from each functional evaluation; or adopt a complicated formalism, as in the FP system of Backus [1].

These problems can be avoided by discarding the usual textual representation of programs and using instead a graph in which arcs carry values normally passed using variables. The source and destinations of an item of data are therefore made obvious: so the user need not keep track of all the occurrences of a variable, or distinguish between similar variable names. An added advantage of such a graphical representation is that potential parallelism is explicit. The first attempt to create such a graphical language resulted in GPL [3], which was never completely developed. A more recent development is the language PROGRAPH [6] which, unlike GPL, provides the modularity that allows programs to be built and debugged in manageable portions, like prgrams in textual languages such as Prolog. A preliminary implementation on a PERQ graphics station is presently undergoing testing. This version is written in Pascal and is extremely complex, particularly the graphics portion, and it is now clear that an alternative implementation language is required. Although in this paper we outline only an interpreter and graphical interface written in Prolog, it is clear that Prolog is ideally suited to implementing the whole PROGRAPH system. This includes the graphical editor, compiler, database manager and operating system.

2. Introduction to PROGRAPH.

In this section we will introduce the main features of PROGRAPH by considering an example, shown in figure 1 at the top of the following page.

A **prograph** consists of a collection of one or more **frames** containing **boxes** connected by **wires**. Data flows through a frame from top to bottom, so wires incident on the top of a box carry inputs to it, while wires incident on the bottom transmit its outputs. Our example shows a prograph consisting of a single frame named REVERSE which reverses a list. This frame contains a **complex** box consisting of two compartments IF and THEN. When a list arrives at the complex box through the top wire, it is passed first to the IF compartment which is activated. The box labelled NONEMPTY receives the list, and outputs either **true** or **false** to the **diamond.** In the first case, the list is passed to the THEN compartment, otherwise it it is passed unchanged to the output of the complex. If the THEN compartment is activated, the list passes to the FIRST-REST box which returns the head and tail of the list on the left and right output wires respectively. The next box to be activated is the one labelled REVERSE, since its output is required as an input to APPEND. The REVERSE box is of course, a recursive call to the frame REVERSE, and produces as output the reverse of the tail of our original input list. The APPEND box then receives this reversed tail on its left input wire, and the head of the original list on its right input wire. It attaches this head to the end of the reversed tail, and passes the result via its output wire to the output wire of the whole complex, which in turn, becomes the output of the frame. The boxes labelled NONEMPTY, FIRST-REST and APPEND are calls to system defined functions. There are a number of other complexes, for example WHILE for iteration. For a more detailed description of PROGRAPH see [6].

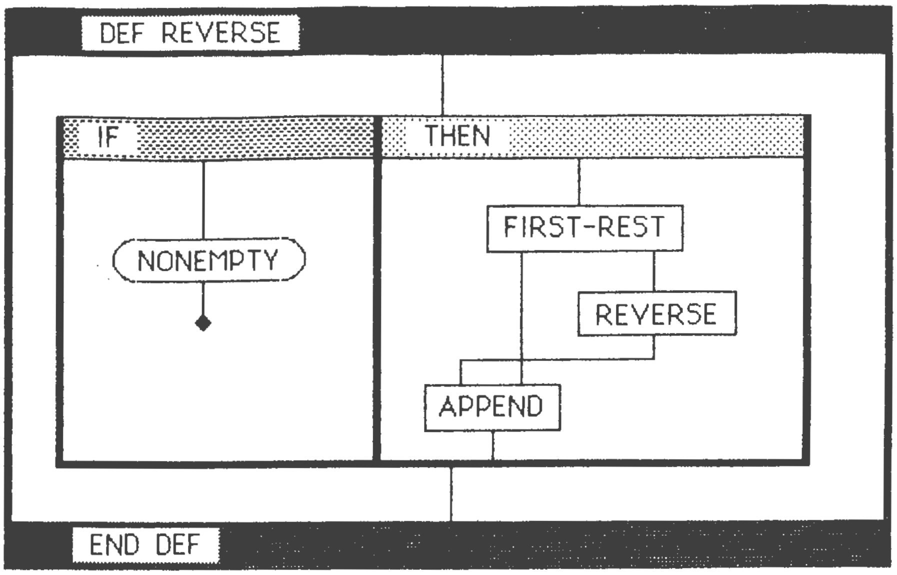


Figure 1: A prograph that produces as output the reverse of the list which is its input.

3. Observing and interacting with the execution of prographs.

It should be noted that figure 1 is a hard copy of the screen representation of a prograph on the PERQ, and that during the execution it is possible to observe various things happening on the screen. When execution begins, the top bar of the REVERSE frame flashes for a brief period, then as data passes down the top wire, a "fireball" rolls down it. Next, the top bar of the IF compartment flashes and a fireball passes to the NONEMPTY box, which in turn flashes and sends a fireball to the diamond. If NONEMPTY produces **true**, the THEN compartment bar will flash, and the process will continue inside the compartment. The recursion produces the following effects on the screen. When the FIRST-REST box has been executed, fireballs travel down its output wires to the APPEND and REVERSE boxes. The REVERSE box then flashes and as a result of the call, the whole process is repeated on the REVERSE frame. Finally an invocation of REVERSE will be reached in which the NONEMPTY box produces **false**, in which case a fireball will appear on the output wire of the output of the complex. This fireball corresponds to an output from the deepest activation of the REVERSE box, so a fireball will appear on the output wire of that box and travel to APPEND, which will flash and produce a fireball which again becomes the output of a REVERSE box. Fireballs will continue to appear from the REVERSE box and APPEND will flash correspondingly until the original invocation of the frame is complete.

Other facilities soon to be incorporated, will allow the user to interact with the interpreter during execution in order to allow easy debugging. In particular it will be possible to execute a prograph in a stepwise fashion, one box at a time. After each box is executed, the user will have the opportunity of asking for values on individual wires, changing these values, and modifying the prograph. It will also be possible to ask that execution proceeds until an error is detected during execution. At that point the system will issue an appropriate message indicating which box is at fault and the nature of the error.

4. Correspondence between PROGRAPH and PROLOG.

As we mentioned in the introduction, implementing PROGRAPH in Pascal was awkward. We will now show that Prolog is the ideal language to use for this purpose. First we show using an example, that prographs correspond to Prolog clauses of a certain restricted type. This correspondence between PROGRAPH and Prolog raises interesting questions about the relationship between functional and logic programming. These issues are currently being investigated. In the following discussion we will use micro-PROLOG [5] since it has some convenient features and is being used for the next experimental version of PROGRAPH.

Below we present a Prolog program equivalent to the prograph in figure 1.

```
((reverse (x) (y))
     (if-then (x) (y)))

((if-then (x) (y))
     (if (x) (true))
     (then (x) (y)))
((if-then (x) (x)))
```

```
((if (x) (y))
    (nonempty (x) (y)))

((then (x) (y))
    (first-rest (x) (z Z))
    (reverse (Z) (Z1))
    (append (Z1 z) (y)))
```

The clause that defines the predicate **reverse** corresponds to the frame REVERSE; this frame has a single input wire, corresponding to the variable **x** and a single output wire corresponding to the variable **y**. The complex box inside the REVERSE frame corresponds to the single literal with predicate **if-then** in the body of the clause. Again the variables **x** and **y** in this literal correspond to the input and output wires of the complex box. Although it is unnecessary from the Prolog point of view, we have divided input variables and output variables of boxes and frames into two lists, the first of which is always the list of inputs. This is because the numbers of inputs and outputs may vary in some PROGRAPH operations, and if inputs and outputs were not distinguished, the Prolog interpreter would not be able to determine which were which, in the corresponding Prolog program.

The clause defining **if** corresponds to the IF compartment of the complex box; its output variable **y** correponds to the wire which terminates in the diamond. We will assume that the predicate **nonempty** implements the PROGRAPH primitive NONEMPTY. Similarly, the clause defining **then** corresponds to the THEN compartment of the complex box, and the literals in the body of this clause correspond in the obvious way with the contents of this compartment. Again, **append** and **first-rest** implement the corresponding PROGRAPH primitives. Note that the order of the literals in the body of this clause corresponds to the order of execution of the corresponding boxes in the prograph. In general there may be more than one possible execution order for boxes, in which case there is more than one possible ordering of literals in a clause.

Clearly we could write a much shorter Prolog program functionally equivalent to the prograph in figure 1; however, the version we have chosen preserves the structural equivalence, which is necessary for the implementation of the graphical features discussed in section 3.

As we mentioned above, Prolog programs corresponding to prographs are of a restricted type. Since each variable in a literal is either an input or an output and inputs are always fully instantiated and outputs are always variables when a literal is executed, every unification is between a variable and a ground term. The only exception is when constants **true** or **false** occur as outputs in a literal for the purpose of selecting alternatives, as in our example. As a consequence these Prolog programs are deterministic.

5. Incorporating graphical information.

In the previous section, we showed that prographs can be translated into directly executable Prolog programs. This translation, however, does not preserve any connection with the display on the screen, so that none of the visual effects described in section 3 can be implemented. In this section, we present two sets of Prolog clauses, one describing the graphical characteristics of the prograph, the other specifying the functional characteristics. This second set of clauses corresponds strictly to the executable Prolog program of the last section, but also includes the necessary links to the graphical information. We will refer to these two sets of clauses as the **graphical** and **functional** bases respectively.

In the following, we refer to the prograph in figure 2, at the top of the next page. This is identical to the prograph of figure 1 except for the addition of integer labels to uniquely identify significant objects.

The graphical base: The following clauses provide graphical information about all the items in the picture except the wires. The predicate name FRAME indicates an object drawn with a banner. In each case, the first parameter specifies further characteristics of the object: for example, **def** indicates the need for a bottom banner, and **rect** indicates a rectangular shape. The second parameter is the identifying integer linking the graphics information with the functional base, presented later. The third parameter is a list of coordinates which locate the object on the screen: c1, c2 etc. would in actuality be pairs of integers. The next two parameters in each clause are lists of identifiers of input and output wires respectively. The last parameter occurring in certain clauses, gives the displayed name of the object. The last clause in this group corresponds to the complex box. The graphical image of this box is actually completely occluded by the images of the IF and THEN compartments; this is reflected in the relationship between their coordinates.

```
((FRAME def 1 (c1 c2) (2) (17) REVERSE))
((FRAME if 3 (c3 c4) (5) ()))
((FRAME then 4 (c5 c6) (9) (16)))
((BOX oval 6 (c7 c8) (5) (7) NONEMPTY))
((BOX diamond 8 (c9) (7) ()))
((BOX rect 10
      (c10 c11) (9) (12 11) FIRST-REST))
((BOX rect 13 (c12 c13) (11) (14) REVERSE))
((BOX rect 15 (c14 c15) (14 12) (16) APPEND))
((BOX complex 18 (c3 c6) (2) (17)))
```

The following group of clauses specifies how the wires are drawn. For brevity we have specified only two of them. In each case the first parameter is the identifying integer, and the second parameter is a list of coordinates specifying how the wire is drawn, starting at the top.

```
((WIRE 2 (c16 c17) ))
((WIRE 14 (c18 c19 c20 c21) ))
```

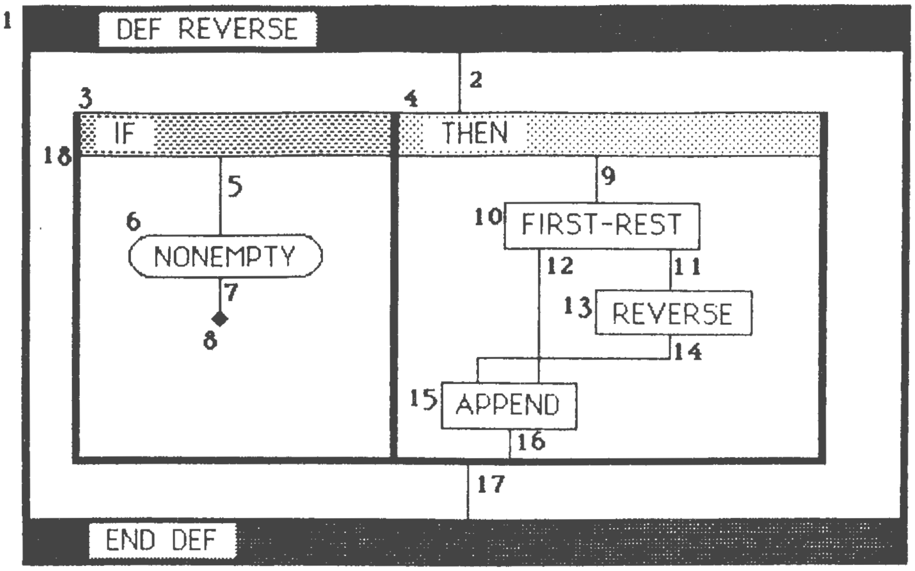


Figure 2: The prograph from figure 1 with integer labels to identify significant objects.

The functional base: In general, every frame, complex, compartment and box corresponds to exactly one literal in the functional base. Similarly, every wire corresponds to exactly one variable, remembering of course that two variables occurring in two different clauses may have the same name. The first parameter of every literal is the identifier of the corresponding object in the prograph: if this parameter is 0, then there is no object in the prograph corresponding to the literal. The first item in the body of each clause is not a literal to be executed, but is a list of pairs relating all the variables occurring in the clause to their corresponding wire identifiers. This list we will call the **wire list**.

```
((reverse 1 (x) (y))
    ((x 2) (y 17))
    (if-then 18 (x) (y)) )

((if-then 0 (x) (y))
    ()
    (if 0 (x) (true))
    (then 0 (x) (y)) )
((if-then 0 (x) (x))
    () )

((if 3 (x) (y))
    ((x 5) (y 7))
    (nonempty 6 (x) (y))
    (diamond 8 () ()) )

((then 4 (x) (y))
    ((x 9) (y 16) (z 12) (Z 11) (Z1 14))
    (first-rest 10 (x) (z Z))
    (reverse 13 (Z) (Z1))
    (append 15 (Z1 z) (y)) )
```

6. The interpreter.

In this section we present the interpreter which executes a prograph by operating on the graphical and functional bases. Note that speed is not of prime importance here since the visual effects are intended as an aid to debugging prographs, so execution should proceed slowly enough for these effects to be observed. Fast execution of a fully debugged prograph can be accomplished by directly executing the corresponding Prolog program, described in section 4. The interpreter obviously requires definitions for primitive operations: it also requires information about the names of these operations. In our example, this information is contained in the following set of clauses:

```
((PRIMITIVE nonempty))
((PRIMITIVE diamond))
((PRIMITIVE first-rest))
((PRIMITIVE append))
```

The interpreter is as follows:

```
((EXEC X Y Z)
     (PRIMITIVE X)
     (X Y Z))
((EXEC X Y Z)
     (CL ((X x Y Z)|X1))
     (FLASH x)
     (FIRE x)
     (EXEC-BODY X1) )
```

```
((EXEC-BODY (x (y z X Y)|Z))
     (QUERY x)
     (FLASH z)
     (EXEC y X Y)
     (FIRE z)
     (EXEC-BODY (x|Z)) )
((EXEC-BODY (x)) )

((FIRE 0))
((FIRE x)
     (FRAME X x y Y|Z)
     (FIREBALL Y))
((FIRE x)
     (BOX X x y z Y|Z)
     (FIREBALL Y))
```

To execute a prograph, a goal literal is supplied of the form:

```
    ?((EXEC <name> <input data list>
            <output variable list>))
```

to which the second definition of **EXEC** is applied since a prograph must consist of a DEF frame. In executing the body of **EXEC**, first **CL** extracts a clause for <name> from the functional base, instantiating **x** with the identifier for the graphical object corresponding to <name>. Next, execution of **FLASH** causes this object to be flashed on the screen, **FIRE** sends fireballs down its input wires, and finally **EXEC-BODY** is called to execute the contents of the frame. The variable **x** in the definition of **EXEC-BODY** is bound to the wire list. **QUERY** allows the user to request the value of a variable by using the cursor to select a wire: the identifier of the wire is found from the graphical base, and its value is located in the wire list. The wires accessible to the user in this way are those inside the frame which is being processed by the current invocation of **EXEC-BODY**. The variable z in the definition of **EXEC-BODY** is bound to the identifier of a box in the frame; this box is **FLASH**ed, then executed by a call to **EXEC**. If the box corresponds to a primitive operation, the first definition of **EXEC** is applied. When execution of the box is complete, **FIRE** sends fireballs down its output wires. Finally, the remaining boxes in the frame are executed by a recursive call to **EXEC-BODY** which receives the wire list **x** as a parameter. When no unexecuted boxes remain, the second definition of **EXEC-BODY** is applied.

Before we explain how **FIRE** and **FLASH** work, we recall that literals in the functional base which have the parameter 0 as graphical identifier do not correspond to any objects in the graphical base. In our example, the literals wires in the clauses defining predicate **if-then** have this property. This is because these clauses perform only a conditional control function. The 0 value inhibits **FIRE**ing and **FLASH**ing. The remaining two definitions of **FIRE** correspond to **FIREBALL**ing input wires of frames and output wires of boxes, respectively.

7. Final remarks.

The above description of the interpreter does not go into any detail about the predicates **FIREBALL**, **FLASH** and **QUERY**. The first two are purely graphical and are largely implemented in assembly language. There are several interesting possibilities for **QUERY**, however; for example, the user could cause the call to **QUERY** to fail, so that backtracking occurs, with the result that wires outside the frame could then be queried. Because prographs are deterministic, interpreting the functional base of a prograph is deterministic. Hence no backtracking occurs unless the user forces it through **QUERY**, in which case, execution will be rolled back exactly one box. Another consequence of this determinism is that the Prolog program corresponding to a prograph can be efficiently executed by an optimising Prolog interpreter.

In a separate report [2], we describe an editor/interpreter for PROGRAPH, also written in micro-PROLOG, that allows prographs to be constructed and tested simultaneously.

References

[1] Backus, J., Can Programming be Liberated from the von Neumann Style?, **Comm. of ACM**, v.21, no.8 (August 1978), 613-641.

[2] Cox, P.T.; Pietrzykowski, T., **Implementing PROGRAPH in Prolog: simultaneous creation and execution of prographs**, Research Report CS8402, School of Computer Science, Acadia University (Jan. 1984).

[3] **GPL Programming Manual**, Research Report, CS Dept., University of Utah (July 1981).

[4] Henderson, P., **Functional Programming: Application and Implementation**, Prentice-Hall, London (1980).

[5] McCabe, F.G.; Clark, K.L., **micro-PROLOG 3.0 Programmer's Reference Manual**, Logic Programming Associates, London (1983).

[6] Pietrzykowski, T.; Bird, L.; Matwin S., **PROGRAPH: A Preliminary Report**, Research Report CS 8302, School of Computer Science, Acadia University (Sept. 1983).

[7] Treleaven, P.C.; et al., Data-Driven and Demand-Driven Computer Architecture, **ACM Computing Surveys**, v.5 (March 1982).

Making "Clausal" Theorem Provers "Non-Clausal"

David L Poole

Logic Programming Group
Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

ABSTRACT

There has recently been a number of papers which extend "clausal" theorem proving systems into "non-clausal" theorem proving systems. Most of these use the justification that the non clausal form eliminates redundancy by not multiplying out subterms. This paper presents the fallacy of such justification by presenting a way to convert to clause form without multiplying out subterms. It also shows how to generate a non-clausal extension to your favourite clausal theorem prover.

1. Introduction

Recently there has been a number of papers which give non-clausal extensions to clausal deduction systems. For example, the extension of resolution to non-clausal form (Manna and Waldinger[80], Murray[82]); the extension of Kowalski[75]'s connection graph proof procedure to non-clausal form (Stickel[82]); and the extension of Andrews[76]' matings to non-clausal form (Andrews[81]).

One of the major disadvantages attributed to clausal form is the need to multiply out subterms. This paper shows that this disadvantage can be overcome by choosing a different algorithm to convert to clause form.

2. Converting to Clausal form - Propositional Case

In this section we will show how to transform a wff in negation normal form, NNF (Andrews[81]) into conjunctive normal form, CNF[†] (or equivalently use Murray[82]'s notion of *polarity* and ignore all explicit negations).

Assume f is a formula in negation normal form (Skolemised, with equivalence and implication expanded out and negations moved in).

If f contains something of the form

$$(\alpha \vee (\beta \wedge \gamma))$$

then it is not in conjunctive normal form and the usual way to convert to conjunctive normal form is to multiply out subterms, viz:

$$((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$$

Thus forming two copies of the subformula α. This causes a problem in theorem proving systems, as a large number of such transformations produces an exponential growth of subterms.

We instead form f^0 which is f with $(\alpha \vee (\beta \wedge \gamma))$ replaced by $(\alpha \vee p)$ where p is a unique atom (not appearing in f). Then form $f' = f^0 \wedge (\neg p \vee \beta) \wedge (\neg p \vee \gamma)$

Theorem: Repeated use of this transformation from f to f' (assuming associativity and distributivity of $\wedge$ and $\vee$) will convert a formula from NNF to CNF.

Proof: (1) the only way a NNF formula will not be in CNF is if it has a subexpression of the form $(\alpha \vee (\beta \wedge \gamma))$ in which case the transformation can be repeated.

(2) the number of $\wedge$'s within the scope of $\vee$'s is reduced by (at least) one each time, thus repeated use of the transformation will terminate.

Theorem: There is no multiplication of subterms in this conversion.

Proof: This is shown by noting that there is only one occurrence of each of α, β and γ in the resulting formula. The repeated p is only an atom and has no structure.

Theorem (Correctness): This transformation preserves the unsatisfiability of the resulting formula.

Proof follows directly from the following Lemma.

Lemma: f is satisfiable if and only if f' is satisfiable.

Proof:

1. (Only if Case) - Suppose f is satisfied by interpretation I. We can assume, without loss of generality, that the denotation of p does not occur in the domain of I. If it does then it can be removed, creating an interpretation still satisfying f.

There are two cases to consider:

a) $(\beta \wedge \gamma)$ is true in I. In this case make $I' = I \cup \{p\}$. I' satisfies f^0 as none of the truth values have changed. (We have substituted a true value for a true value). I' satisfies $(\beta \wedge \gamma)$ so it satisfies $((\neg p \vee \beta) \wedge (\neg p \vee \gamma)$ so I' satisfies f'

b) $(\beta \wedge \gamma)$ is false in I. In this case make $I' = I \cup \{\neg p\}$. Then I' satisfies f^0 as none of the truth values has changed. $\neg p$ is true in I' so $((\neg p \vee \beta) \wedge (\neg p \vee \gamma))$ is true in I', so I' satisfies f'.

2. (If Case) - Suppose f' is satisfied by interpretation I'. Then in particular both f^0 and $((\neg p \vee \beta) \wedge (\neg p \vee \gamma))$ are true in I'. There are two cases to consider:

a) p is true in I'. Then as $((\neg p \vee \beta) \wedge (\neg p \vee \gamma))$ is true in I', $(\beta$ and $\gamma)$ must be true in I', so f is true in I' as none of the truth values have changed.

[†] If you would rather convert to disjunctive normal form then read the dual of the paper (swap $\wedge$ and $\vee$; swap *true* and *false*; and read *valid* for *unsatisfiable*).

b) p is false in I'. In this case f must be true in I' as replacing something that was false in a conjunction or a disjunction cannot make the conjunction or disjunction false when it was previously true. Note that there are no negations to be considered, as all negations are moved in, and p does not involve a negation in f.

Q.E.D.

3. The Predicate Calculus Case

The predicate calculus conversion is like the propositional calculus case except that the new atomic formula introduced is of the form $P(z_1, \cdots, z_n)$ where $z_1, \cdots, z_n$ are the free variables in $(\beta \wedge \gamma)$ and P is a unique n-place predicate symbol. Let f' be created from f in the same way as for the ground (propositional) case.

Theorem: For the predicate calculus case, f is unsatisfiable if and only is f' is unsatisfiable.

Proof:

Case . 1: Suppose f is satisfied by I. Define $P(z_1, \cdots, z_n)$ to be true in I' in exactly those cases for which $(\beta \wedge \gamma)$ is true in I. Then for each of the values for the variables $z_1, \cdots, z_n$ the same argument as for the ground case holds. So f' is satisfied by I' for all values of $z_1, \cdots, z_n$

Case 2: Suppose f' is satisfied by I' Then for each value of $z_1, \cdots, z_n$ the ground argument holds. So f is satisfied by I'.

Q.E.D.

4. Using the Transformation

4.1. What Has Been Gained?

The gain that occured is that there is only one copy of α in the resulting formula. If α is a large structure, then once α has been proven false (or resolved away) once then both β and γ can be used. In the distributive form, α must be resolved away for both β and γ.

If n is the number of $\wedge$'s in the scope of $\vee$'s, then in the transformation here there are $2n+1$ clauses produced. In the traditional transformation there may be 2^n clauses produced. (Consider the case of converting something in disjunctive normal form (two literals per conjunct) into conjunctive normal form.)

4.2. Making the Transformation Implicit

The disadvantage of such a transformation may be in the cost of creating the new literals. In this section we show how to avoid creating new literals, and how to avoid doing the explicit transformation at all. The first approach is to change the deduction system to make the transformation implicit as a special case. The second is to modify the preprocessing that has to be carried out before the (unchanged) deduction system runs.

As an example of the former, consider a resolution-type theorem prover. In the transformed system, the only atoms that p can unify with are the instances of p explicitly created in the transformation. In particular, only three instances of the atom p appear. If p is ever successfully resolved away then both α and either β or γ is resolved away. If α is resolved away, then instead of leaving the residual literal p, and letting it resolve with one of the clauses $(\neg p \vee \beta)$ or $(\neg p \vee \gamma)$, and producing β or γ to be resolved away, the deduction system can be modified to recognise this case and produce β or γ one step earlier. The other case of having resolved away one of β or γ, leaves $(\alpha \vee p)$ as the only choice to resolve away p. Instead of having p explicit, the theorem prover can try immediately to resolve away α.

In connection graph proof procedures, the connection graph contains all of the information about unifications. In particular, after the connection graph is built, the internal form of the literals is irrelevant. A connection graph builder, instead of creating the p's and then adding the two connections, and forgetting about the internal forms of the p's, can build the connections without creating the p's at all.

5. Conclusion

This paper demonstrates that one of the advantages that "non clausal" theorem proving has over "clausal" theorem proving is **not** that converting to clause form multiplies out subterms.

In any theorem proving method the only unifications with the introduced atomic symbol will be those given by the procedure above. Therefore the effect of the connection can be calculated before any actual deduction, so the above procedure may not need to be carried out at all.

If you like the idea of non-clausal theorem proving then find your favorite clausal theorem prover; allow input in non-clausal form; find a way to do the transformation above implicitly; and you have an extension of the theorem prover to the non-clausal case.

6. References

Andrews,P.B.[76],"Refutation by Matings", *IEEE Trans. Comput. C-25.* pp 801-807.

Andrews,P.B.[81],"Theorem Proving via General Matings", *Journal A.C.M.* Vol 28, No 2, pp 193-214.

Kowalski,R.[75],"A Proof Procedure Using Connection Graphs", *J.A.C.M.* Vol 22, No 4, pp 572-595.

Manna,Z.and Waldinger,R.[80], "A Deductive Approach to Program Synthesis", *A.C.M.T.O.P.L.A.S.* Vol 2, No 1 pp 90-121.

Murray,N.V.[82],"Completely non-clausal theorem proving", *Artificial Intelligence,* Vol 18, No 1, pp 67-85.

Stickel,M.E.[82],"A nonclausal connection-graph resolution theorem-proving program", *AAAI-82,* pp 229-233.

Logic As An Interaction Language

M.H. van Emden

Department of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1
Canada

ABSTRACT

This paper addresses the problem of interactive input and output in logic programming. We propose ***incremental queries*** as one method for declarative interactive I/O. As another method we propose a new model of computation (the ***query interaction model***) based on Sergot's "Query-The-User" expert system interface.

1. Introduction

Prolog contains a well-defined declarative subset, which does not, however, provide for input or output. Hence an important advance is to achieve declarative I/O in a logic programming language. In this paper we propose two ways of modifying Prolog that make its I/O declarative: incremental goal statements and the query-interaction model. This last proposal is merely to carry the idea of Sergot's "Query-The-User" [2] a step further.

2. Incremental goal statements

From the mathematical viewpoint, the result of a logic program computation is a substitution to be applied to a goal statement. Declarative output is some display of this substitution. A convenient form is the one used in the SIMPLE front end to micro-Prolog [1], where one can request for output an arbitrary list typically containing some of the goal statement's variables. On successful proof of the goal statement the list is displayed with the substitution applied to it.

We propose to make it possible to submit queries in installments rather than all at once.

A non-incremental query has the form:

$$\text{which}\ (\ <\text{pattern}>\ : G_1\ \&\ G_2\ \&\ \ldots)$$

Before the colon is an *answer pattern:* any expression containing, say, the variables $x_1 \cdots x_m$. After the colon is a *goal statement:* a conjunction of atomic formulas typically containing these variables. The effect of this query depends on whether some instance of the goal statement is a logical consequence of the current set of assertions. If this is the case, then the effect is to display such an instance substituted into the answer pattern.

Now we can imagine a system where not just the final instance of the answer is displayed, but also all its intermediate instances: the one after G_1 has been proved, after $G_1\ \&\ G_2$ has been proved, etc. In general, the initial segments $G_1\ \&\ \cdots\ \&\ G_i$ may have more than one answer substitution; the first one found is not necessarily compatible with those of later initial segments.

Let us first consider the special case of deterministic goals. Here it may be advantageous if the user can postpone typing in G_{i+1} till after he has seen the substitution for $G_1 \& \ldots \& G_i$. This is the idea of an ***incremental query.*** We present it as a generalization of the existing "which" query, which can coexist with incremental queries. A "which.." query does not complete a query but only adds the next increment to the query currently being built up. Thus the sequence of incremental queries

$$\text{which..}(x_{11}\ \cdots\ : G_{11}\ \&\ \ldots)$$

$$\text{which..}(x_{21}\ \cdots\ : G_{21}\ \&\ \ldots)$$

$$\cdots$$

$$\text{which}(x_{n1}\ \cdots\ : G_{n1}\ \&\ \ldots)$$

causes the same goal statement to be proved and the same substitutiom to be determined as

$$\text{which}(y_1\ \cdots\ : G_{11}\ \&\ \cdots\ \&\ G_{21}\ \&\ \cdots\ \cdots\ \&\ G_{n1}\ \&\ \ldots)$$

where $y_1 \cdots$ is a pattern containing the union of the x-variables.

An incremental query allows interaction: the $(i+1)$st "which.." subquery can be determined by means of the substitution displayed as a result of the first i increments. Note that only a "which" completes the query: a variable name in different "which.." expressions not separated by a "which" names the same variable. In programming language terminology, a first "which.." opens a block; every "which" closes one.

We show an example interaction by means of incremental goal statements. Let "trans" denote a relation with database states as first and third arguments and with a transaction as second argument. This relation holds if and only if the third argument is the result of applying the second to the first "init" is some standard initial database state.

$$\text{which..}(x1 : \text{trans(init add(red(cherry))}\ x_1))$$

The effect of this subquery is to display the value of x_1, viz. x_1 = {red(cherry)}.

$$\text{which..}(x_2 : \text{trans}(x_1 \ \ \text{add(green(apple))} \ \ x_2))$$

and x_2 is now the database state resulting from adding both red(cherry) and green(apple) to init. In this way states are built up by successive query increments containing chained "trans" goals. Each time the database is displayed. The next query increment can take into account the state last displayed.

Suppose we regret the transaction add(green(apple)) and that we now enter the query increment

$$\text{which..}(x_2 : \text{trans}(x_1 \ \ \text{add(red(apple))} \ \ x_2))$$

Here it seems reasonable to implement trans in such a way that its third argument always has to be an uninstantiated variable. Then the last query increment will be voided by a control error, signalling relation misuse causing an implementation exception. Otherwise failure will result: no x_2 exists as required by the last two query increments. But let us assume that a control error occurs so that the first two query increments still stand. The user's mistake in the voided query increment can now be corrected:

$$\text{which..}(x_3 : \text{trans}(x_1 \ \ \text{add(red(apple))} \ \ x_3))$$

We see that because we have names of previous states, we can undo transactions.

If we are not interested in being able to undo, and only use trans in chained mode, then the name of states are not needed. For such situations a specialized notation can be used. Instead of

$$\text{which..}(x_1 : \text{trans(init} \ <\!\text{tr}-\text{ac}-1\!> x_1))$$

$$\text{which..}(x_2 : \text{trans}(x_1 \ \ <\!\text{tr}-\text{ac}-2\!> x_2))$$

...

we could write something like

$$\textit{chain} \ \text{trans init}$$

$$<\!\text{tr}-\text{ac}-1\!> \qquad \{x_1 \text{ is displayed}\}$$

$$<\!\text{tr}-\text{ac}-2\!> \qquad \{x_2 \text{ is displayed}\}$$

...

With this convention the interaction looks completely imperative. But we know that it can be syntactic sugar for declarative I/O using incremental queries. For example, the states and transactions could be those of the BASIC programming language. Then our proposal for the sugared form of query increments containing chained "trans" goals results in an interaction indistinguishable from a conventional one in BASIC. Yet ours is (sugared) logic. However, we do not advocate this usage because BASIC computations can usually be recast as more problem-oriented relations not involving BASIC states, so that not only incremental queries are inappropriate but also the entire state-change paradigm.

It has been suggested by Marek Sergot (personal communication) that incremental queries are useful for nondeterministic goals as well. He suggests the example of a logic program constructing a timetable satisfying constraints expressed as goals of a query. The user enters a query containing all constraints he can think of. In response, the program displays the first solution. The user may then notice some undesirable feature of this solution. If this first query were incremental, the user can then continue it with a goal further constraining the solution to avoid the objectionable feature. In this way the user can converge interactively to a satisfying timetable without having to be able to have in mind in advance all the required constraints.

Robert Kowalski (private communication) has looked at incremental goal statements from the metalevel. From this august vantage point they look a very special case indeed: he remarked that the successive goal statements can be regarded as a sequence of independent queries having the special property that each is an extension of the previous one. That immediately suggests a more general mechanism allowing one to generate sequences of object-level queries specified by a logic program at the metalevel. Such a program could of course generate any sequence. However, if it would generate the next element of the sequence by tacking on a goal to the previous element and if it would ask the user (see below) what this additional goal should be, then Kowalski's general mechanism would simulate the proposed incremental goal statements.

3. The query-interaction model

We introduce the query-interaction model by a brief sketch of the main idea of "Query-The-User" [2]. Prolog programs consist of conditional assertions, which can be regarded as rules, and of unconditional assertions which are more like database facts. If a query matches an unconditional assertion it is answered immediately, otherwise it may match a conditional assertion resulting in a modified query. Ultimately, however, all queries must be reduced to ones that can be answered by a database entry. Conventional Prolog execution causes a query to fail if every way of choosing matching assertions leaves at least one unmatchable query.

In Query-The-User this last situation is regarded as a symptom of lack of information for which there exists, potentially, a remedy. The user is regarded as an extension of the program database. Whenever missing information is encountered, the user is queried. Here we emphasize two aspects of Query-The-User. The first is that there is a new type of relation between user and machine: it is symmetrical in the sense that queries can go both ways. The second is that, because of this symmetry, declarative *input* has been added to the declarative output of microProlog's SIMPLE. Moreover the declarative I/O thus achieved is *interactive*.

Logic programming has inherited some aspects of the legacy of conventional programming: solipsistic program execution is the norm; I/O is tacked on in the form of certain system functions. Kowalski's procedural interpretation of logic, where definite clauses are interpreted as procedure declarations and where SLD resolutions are interpreted as procedure calls, is based on this solipsistic model of computation. Prolog's I/O by means of procedures with side effects is a part of this inheritance.

In Query-The-User, logic programming has contributed a new model of computation; let us call it the *query interaction model*. The operational interpretation of a definite clause

$$A \leftarrow B_1 \,\&\, \cdots \,\&\, B_n$$

is that of a ***query reflector:*** when a query of the form

$$A$$

is directed at it, this results in queries

$$B_1 , \cdots , B_n$$

being reflected from it, without any ***a priori*** commitment as to where these reflected queries go. Let us consider as possible destinations of reflected queries

A) the user

B) the clause set itself

C) other clause sets (in the same machine or not)

D) programs in other languages (in the same machine or not)

At the solipsistic end of a spectrum of possibilities, the destination is always B. But even conventional Prolog implementations are not that extreme: if the predicate symbol of a reflected query belongs to a certain class, then its destination is D. This class is usually called "built-in"; these queries typically ask for arithmetical operations, I/O by side effect, deviation from the normal control regime, type conversion, and perhaps other facilities. At the interactive extreme of the spectrum, the destination is never B.

We see that deciding where reflected queries are answered now becomes a ***new dimension of control.*** In the procedural interpretation of logic, control is needed to determine in what order goals are selected and in what order matching clauses are tried for a given selected goal. These decisions also have to be taken in the query interaction model. But there one has the additional choice in determining the destination of reflected queries.

Consider an existing system from the point of view provided by the query interaction model. APES (see Hammond's contribution in [1]) determines the destination of reflected queries in the same way as conventional Prolog interpreters: by determining to which class the predicate symbol belongs. APES goes beyond conventional Prolog in allowing the user to determine these classes by an interaction at the meta-level. Another control strategy would be always to try first the rule set itself and to send the reflected query elsewhere in case of failure, trying perhaps first option D and then option A. This gives quite different properties: to make the query interaction model practically attractive, it needs to be implemented with a flexible and convenient facility for the user to specify this new aspect of control.

4. Acknowledgements

I have greatly benefited from the hospitality of the Logic Programming Group at the Department of Computing in Imperial College, University of London. Conversations with Peter Hammond, Robert Kowalski, Marek Sergot, and David Weir were especially valuable. This paper was written in connection with a project sponsored by the Data Systems Division of the International Business Machines Corporation, who kindly gave permission for its publication.

5. References

[1] F.G. McCabe and K.L. Clark, Micro-Prolog: Programming in Logic, Prentice Hall, 1984.

[2] M. Sergot: A Query-The-User facility for logic programming. Proceedings of the European Conference on Integrated Computing Systems, P. Degano and E. Sandewall (eds.), North Holland, 1983.

ROG-O-MATIC: A Belligerent Expert System

Michael Mauldin, Guy Jacobson, Andrew Appel and Leonard Hamey
Computer Science Department, Carnegie-Mellon University,
Pittsburgh, PA 15213

Abstract

Rog-O-Matic is a novel combination of algorithmic and production systems programming techniques which cooperate to explore a hostile environment. This environment is the computer game **Rogue**, which offers several advantages for studying exploration tasks. This paper presents the major features of the Rog-O-Matic system, the types of knowledge sources and rules used to control the exploration, and compares the performance of the system with human Rogue players.

Introduction

Rog-O-Matic plays the computer game Rogue, wherein the object is to explore and survive a complex and hostile environment. Rogue is an example of an *exploration task*, *ie* given an undirected planar graph, a starting node, and a visibility function which maps each node into a subset of visibile nodes, exploration entails traversing edges so as to see all of the nodes. Minimizing the number of nodes visited is a subgoal. Studying exploration requires two things: terrain to be explored, and an explorer to search it. There are four major advantages for choosing Rogue as a terrain generator:

- Success in Rogue depends heavily on successful exploration.
- It is a standard game, designed for human play.
- It has an objective, scalar measure of performance (*i.e.* the score).
- There is an abundance of human volunteers to calibrate the performance measure.

The Rogue exploration task is complicated by adversaries which attempt to prevent the explorer from reaching his goal. Carbonell has studied the problem of planning in an adversary relationship [2], but planning in Rogue is hampered by the randomness of the adversary. Where the probabilities are known, search trees can be built with each possible next state labelled with its transition probability. The action with the highest expected value can then be selected [1]. In Rogue, however, these probabilities are unknown to the player, and can change from game to game. *Scenarios* have also been used to analyze combat situations [9], but when unseen opponents can appear at any time, and when many of the combat parameters are unknown, choosing the right scenarios can be very difficult.

Rog-O-Matic is designed as an knowledge based or "expert" system because a search based solution is inherently intractable. There are as many as 500 possible actions at any one time; for each action there are several thousand possible next states.

Rog-O-Matic differs from other expert systems in the following respects:

- It solves a dynamic problem rather than a static one.
- It plans against adversaries.
- It plays a game in which some events are determined randomly.
- It plays despite limited information.

In this paper we introduce the Rogue environment and discuss the architecture, knowledge sources, and production rules used by Rog-O-Matic to explore that environment. We also discuss the system's implementation and compare its performance with that of human Rogue experts. For more a more detailed description of the program, see [7].

The Rogue Environment

The Rogue game is described in more detail in [8]; an overview is given here. The environment is a cave composed of levels; each level contains between six and nine rectangular rooms connected by tunnels. The game player takes the role of explorer, searching for gold and fighting the monsters which guard the cave. Magical items such as potions or weapons are also scattered about, and these can be used to advantage in combat. Traps are hidden throughout the rooms to thwart careless players.

The object of the game is to explore to the 26th level, find the *Amulet of Yendor*, and bring it back to the surface. A player who achieves this goal is called a *Total Winner*. As the explorer goes deeper into the cave, the gold becomes more plentiful and the monsters become more difficult to defeat. As the explorer kills monsters in combat, he becomes a better fighter. On the average, monsters increase in ferocity faster than the player increases his skill, so reaching the 26th level is almost impossible. The player's score is the amount of gold he obtains, with bonuses given for retrieving the amulet.

The Expert System Architecture

Rog-O-Matic is a combination of algorithmic knowledge sources and production rules. Where uncertainty is low, algorithmic methods provide fast calculation of relevant information. Where uncertainty is high, heuristic production rules provide reasonable behavior. Knowledge sources and production rules interact through a world model or "blackboard" [3]. Because the system is implemented in an imperative language, the production rules are statically ordered. Dynamically ordered rules would have been much more complicated to code, and very few of the rules would benefit from this added complexity.

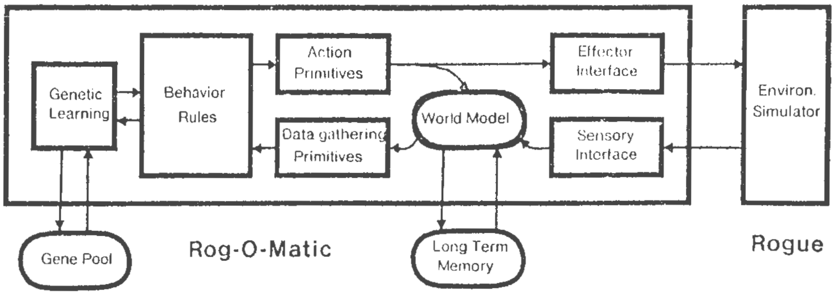


Figure 1: Rog-O-Matic System Architecture

Figure 1 shows the Rog-O-Matic high-level system architecture. Rog-O-Matic plays the game by intercepting characters from the terminal stream, and sending command characters back to the Rogue process. The input interface must convert a stream of characters designed to draw a meaningful picture on a CRT screen into a representation of the world state. This function is performed by the *sensory* module. The less difficult task of converting motion and action directives into Rogue commands is done by the *effector* module, which is shown mainly for symmetry. Langley *et al* have described the advantages of simple sensory/effector interfaces in studying a reactive environment [6]. The long term memory file is a portion of the world model which is saved between games. The gene pool tracks successful settings of parameters which modify the behavior rules. All long term learning is achieved by modifying these two files.

Knowledge Sources

The world model is operated on by a variety of knowledge sources, which take low level data from the world model and process it into higher level information. Eventually, the information is encoded at a sufficiently high level for the rules to operate directly on it. The action primitives also change the world model. This feedback permits the production rules to encode inference mechanisms and short term memory. A partial list of knowledge sources is given here:

- Sensory system (**sense**) Builds low level data structures from Rogue output.
- Object map (**objmap**) A data structure which tracks the location and history of objects in the environment (such as weapons or monsters).
- Inventory handler (**invent**) A database of items in Rog-O-Matic's pack.
- Terrain map (**termap**) A data structure recording the terrain features of the current level.
- Connectivity analyzer (**connect**) Finds cycles of rooms (loops).
- Path calculation (**pathc**) Performs weighted shortest path calculations.
- Internal state recognizer (**intern**) Tracks the health and combat status of Rog-O-Matic.

Production Rules

The rules are grouped hierarchically into experts; each expert performs a related set of tasks. These experts are prioritized in order of estimated contribution to survivability. For example, if the melee expert decides that a monster must be fought, that decision overrides the advice of the object acquisition expert calling for an object to be picked up. If the melee expert suggests no action, then the object acquisition expert's directive is acted upon. The basic structure resembles a directed acyclic graph (DAG) of *IF-THEN-ELSE* rules. Figure 2 shows the information flow between these experts. Here is a list of the most important experts:

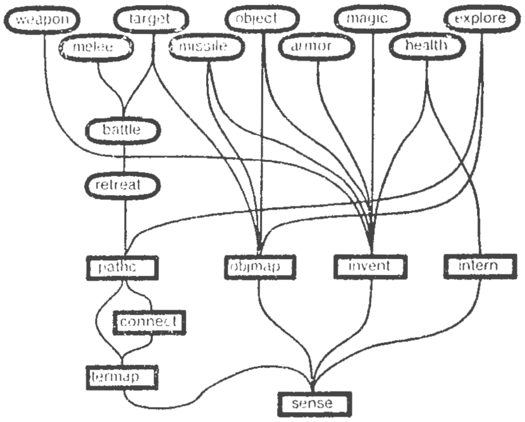


Figure 2: Experts (ovals) and Knowledge Sources (boxes)

- Weapon handler (**weapon**) Chooses weapon to wield.
- Melee expert (**melee**) Controls fighting during combat.
- Target acquisition expert (**target**) Controls pursuit of targets.
- Missile fire expert (**missile**) Fires missiles (arrows, spears, rocks, *etc.*) at distant targets.
- Battlestations expert (**battle**) Performs special attacks, initiates retreat.
- Retreat expert (**retreat**) Uses the **termap** and **connect** to choose a path for retreat.
- Object acquisition expert (**object**) Picks up objects.
- Armor handler (**armor**) Chooses armor to wear.
- Magic item handlers (**magic**) Manipulates magic items.
- Health maintenance (**health**) Decides to eat when hungry and to heal damage by resting.
- Exploration expert (**explore**) Chooses next place to explore, and controls movement.

Algorithmic Knowledge Sources

Of the algorithmic knowledge sources, the path calculator is the most interesting. It reads the terrain map and determines weighted shortest paths from Rog-O-Matic to the nearest location satisfying specified criteria (such as the nearest

```
/*
 * If we can die in one turn and we are not at peak
 * strength, we might want to retreat.  Don't try to
 * run if we are confused or being held by something.
 * If we are on a door, wait until the monster is next
 * to us (that way we can shoot arrows at him, while
 * he catches up). Don't run away from Dragons!!!
 * They'll just flame you from behind.
 */

if ((die_in (1) && Hp+Explev < Hpmax) &&
    !confused && !beingheld &&
    (!on(DOOR) || turns < 1) &&
    !streq(monster, "dragon") &&
    runaway ())
{ display ("Run away! Run away!"); return(1); }

/*
 * We can't run and are next to a monster which can kill
 * us in one turn.  Read a scroll of teleportation.
 */

if (die_in (1) && turns == 0 &&
    (obj=havenamed (scroll, "teleportation")) >= 0 &&
    reads (obj))
{ return (1); }
```

Figure 3: Sample Rules from the *Battlestations* Expert

unknown object, or the nearest escape route). The edge costs are small, bounded integers which encode known or possible dangers (such as traps or unexplored squares). A breadth-first search explores outward from the current location, and any square with a non-zero cost (an *avoidance* value) has its cost decremented and then square is placed unexamined at the end of the search queue. The result is a weighted shortest path obtained in time linear to the number of terrain squares. Since exploration requires the system to spend most of its time moving, a fast path calculator is essential.

Most of the actions taken by Rog-O-Matic are simpler than movement, and these actions are performed directly by the production rules. For example, when the *melee expert* has determined that a battle is underway, it puts certain key parameters of the battle, such as the strength of the monster, the number of turns which the monster will require to reach the player, and its direction, into the blackboard and invokes the *battlestations* expert. *Battlestations* decides whether to attack with the weapon currently in hand, attack with a special magic item, or to retreat. Figure 3 shows some sample rules extracted from *battlestations.*

Learning

There are three kinds of learning in Rog-O-Matic: *short term*, for object recognition, *long term* for monster characteristics, and *genetic learning*, for parameter adjustment. Short term learning is used to gather information about the offensive and defensive weaponry available to the player. Proper use of this weaponry is crucial to success in Rogue. At the beginning, the player does not know the characteristics of the majority of the objects in the game. Knowledge about items can be gained through experimentation, or by using the various *Identify* scrolls in the game. This kind of learning is closed, since there are fewer than 50 different things any object can be. A hand-coded database is used to match the results of experimentation (*ie* trying an item to see what it does) with the nature and use of that item.

A more interesting problem is learning about monster characteristics. When deciding whether to attack or retreat, an accurate estimate of the opponent's strengths and weaknesses is needed. A hardwired table of these estimates was used in early versions of Rog-O-Matic, but the monsters were changed in the latest release of Rogue (version 5.3). Rather than build another table by hand, we added a learning package which estmiates the offensive and defensive strength of each monster encountered. These values are retained from game to game, and are stored in a long term memory file. This kind of learning is not intended to increase performance (since the earlier versions already had perfect information about monster characteristics), but to provide flexibility for the program. Changes in the opponents do not require changes in the program.

The most recent addition to the program is a genetic learning component based on the genetic adaptive algorithm of Holland [5]. This module adjusts the following seven parameters which control the program's high-level behavior:

- when to search for traps
- when to search for secret doors
- when to rest (to heal damage)
- when to shoot arrows
- when to experiment with items
- when to retreat
- when to attack sleeping monsters

Twenty different settings of these parameters are kept in another long term memory file called the *gene pool.* The average score of each such *genotype* is also retained. High scoring genotypes are crossed to generate new genotypes, and the resulting parameter settings are evaluated until one genotype is significantly worse than the others, at which point it is discarded and another new genotype is generated. This learning process produced versions of the program with average scores twice as high as the best versions without genetic learning (mean scores greater than 1500 as opposed to 700 for older versions, playing against Rogue 5.3).

Implementation

Rog-O-Matic runs under the Berkeley 4.2 Unix operating system. It is composed of 12,000 lines of C code; C was chosen for convenience and speed. It offers direct access to the operating system primitives required to interface to the Rogue game (a necessity, since we wanted our program to play exactly the same game as the human players). C code also runs much faster than most production systems languages, and this speed was necessary for Rog-O-Matic to play enough games for its performance to be measured. The program takes about 30 CPU seconds (generating about 400 actions) to explore one level, and Rogue takes an additional 15 CPU seconds calculating its part of the simulation. At CMU, Rog-O-Matic has played more than 12,000 games in two years on one VAX 11/780. Rog-O-Matic is also running at more than 40 other installations in the US, Canada, and England.

Performance

Evaluation of expert systems can be a difficult problem [4], but two measures of a game playing program are obvious: its reputation among its competitors and its raw scores. Rog-O-Matic has garnered a very favorable reputation as an expert Rogue player, and many human players have reported learning

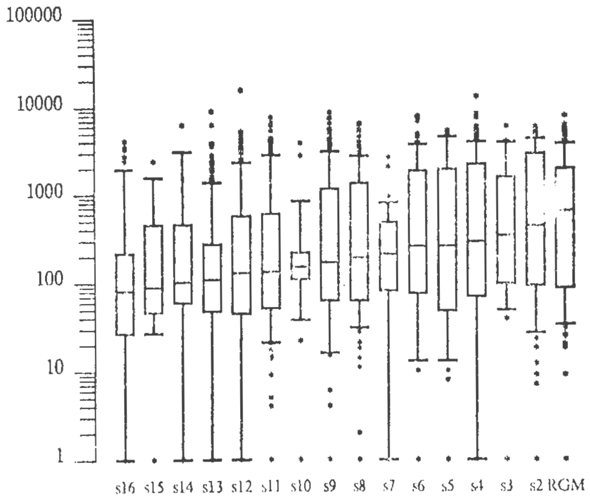


Figure 4: Log Scores *vs.* Player, Sorted by Median

a lot about Rogue both by watching it play and by reading the source code.

We compared Rog-O-Matic scores with 15 of the best Rogue players at CMU over a two month period, and Rog-O-Matic played as well as the human experts. Figure 4 shows percentile plots of these games, sorted by median score. Rog-O-Matic obtained the seventh best high score and the best median score of any player. Rog-O-Matic has also been a *Total Winner* playing against each major version of Rogue.

The graph includes all players with ten or more games against Rogue (version 5.2) on the GP-Vax at CMU. The data were collected over a two month period from January 1, 1983 to February 21, 1983. Boxes are drawn from the lower quartile to the upper quartile, with a line at the median. The whiskers are drawn at the 10th and 90th percentile marks, and extreme games are plotted as asterisks. The vertical scale is the score (drawn logarithmically) and the horizontal scale is the player's rank. The vertical scale is drawn logarithmically because Rogue scores approximate a log-normal distribution.

Problems

The static ordering used prevents some Rogue tactics from being cleanly represented. An example is the use of armor. In certain cases the best armor should be worn, and in other cases the second best armor should be worn. The rule calling for the best armor must be disabled while the second best armor is in place, and yet be reactivated at the right time. Each of these dependencies requires a new global variable in short term memory. Dynamic rule ordering might be a cleaner way to solve this problem.

Another problem is single-mindedness. For example, when retreating from a monster, it is often possible to pick up objects on the way to an escape route. Rog-O-Matic does not consider that a single action may satisfy multiple goals, so the program fails to take advantage of such situations. In other cases the presence of a secondary factor requires more creative actions than are currently possible. This problem is most severe when Rog-O-Matic has to combat multiple monsters. Its heuristic is to fight the most dangerous monster first, but there are situations in which this rule fails miserably. In Rogue, failing miserably usually results in death.

Conclusion and Future Work

By combining efficient algorithmic knowledge sources with a rule based control mechanism, we have produced a program demonstrably expert at the game of Rogue. The result is a system capable of performing exploration tasks while also fulfilling the goal of self-preservation. The system can function well in the face of uncertain dangers, make well considered *fight or flight* decisions, and retreat successfully when it faces overwhelming opposition. Short term and long term learning have provided sufficient flexibility to handle a changing environment and reduced the programming time required to handle newer versions of Rogue. Genetic learning has resulted in higher average scores. Our current work is focused on increasing the number of parameters available for tuning by the genetic learning algorithm, with the goal of increasing the program's high scores.

Acknowledgments

We would like to thank the Rog-O-Matic user communities at both CMU and across the ARPANet for their thousands of hours of testing, and also the many adventuresome Rogue players who provided thousands of data points on human Rogue performance. Thanks to Damon Permezel, of the University of Waterloo and Dan Reynolds, of the University of Texas at Austin, for running the first Rog-O-Matics to be total winners against Rogue 5.2 and Rogue 5.3, respectively. And of course, special thanks go to Ken Arnold and Michael Toy for creating a very interesting (and very, very hostile) environment.

References

[1] H. Berliner, "Experiences in Evaluation with BKG—A Program that plays Backgammon," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence,* 1977.

[2] J. Carbonell, "Counterplanning: A Strategy-Based Model of Adversary Planning in Real-World Situations," *Artificial Intelligence,* Vol. 16, 1981.

[3] L. Erman, F. Hayes-Roth, V. Lesser and R. Reddy, "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty," *Computing Surveys,* Vol. 12, No. 2, June 1980.

[4] J. Gaschnig, P. Klahr, H. Pople, E. Shortliffe, and A. Terry, "Evaluation of Expert Systems: Issues and Case Studies," in *Building Expert Systems,* F. Hayes-Roth, D. Waterman, and D. Lenat, ed., Addison-Wesley, Reading, Mass., 1983.

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

[6] P. Langley, D. Nicholas, D. Klahr and G. Hood, "A Simulated World for Modelling Learning and Development," *Proceedings of the Third Annual Conference of the Cognitive Science Society,* 1981.

[7] M. Mauldin, G. Jacobson, A. Appel and L. Hamey, "ROG-O-MATIC: A Belligerent Expert System," Tech. report CMU-CS-83-144, Carnegie-Mellon University, July 1983.

[8] M. Toy and K. Arnold, "A Guide to the Dungeons of Doom," Unix Documentation, Department of Electrical Engineering and Computer Science, University of California,.

[9] R. Wall and E. Rissland, "Scenarios as an Aid to Planning," *Proceedings of the second meeting of the American Association for Artificial Intelligence,* 1982.

AN EXPLANATION SYSTEM FOR FRAME-BASED KNOWLEDGE ORGANIZED ALONG MULTIPLE DIMENSIONS

Ron Gershon
Yawar Ali
Michael Jenkin

Laboratory for Computational Medicine
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4

Abstract

This paper describes an explanation system for frame based knowledge about events, as presented in a visual motion expert system. As such, it can be applied to representations that embody different frame organizational relationships such as is_a, part_of, instance_of, similarity, and time. This may be contrasted with most current expert systems which employ rule-based knowledge representations. In addition, such expert systems typically do not deal with complex spatio-temporal information and only a small number of them have any explanation capabilities. The system described in this document is capable of making inferences about frame comparisons and temporal relationships not present in the knowledge base, and provides output in both textual and pictorial formats. The graphical format is particularly useful for revealing the structure of the knowledge frames. The system has been implemented and tested on a knowledge base designed for human left ventricular performance assessment and examples of interaction with the system will be presented.

Introduction

One of the early conclusions of designers of expert systems was that they should have an explanation capability [10]. Although most of the early systems demonstrated interesting results in their domain of expertise, there was a need to see not only correct results, but also the means by which they were achieved. Typically knowledge appeared as rules and was organized through the implicit AND/OR tree formed by the premise-action pairs, and context trees that group rules according to their applicability [15]. Rule-based systems have many benefits; however, their ability to accomodate an explanation facility that deals with abstractions and comparisons of knowledge units seems limited.

In this document, we present an explanation capability which was built upon the ALVEN system [12, 13, 14]. ALVEN evaluates human left ventricular performance, employing an extensive frame-based knowledge base. With its rich representation scheme, which includes different knowledge organizations and frame inter-relationships, it is able to deal with complex issues such as time and space. In the case of ALVEN, abstraction can be achieved with the different knowledge organizations found in the representation, such as generalization and aggregation, while comparison is available between frames connected via similarity links. We concentrated mainly on explanation of the knowledge base. Although the explanation of reasoning was not addressed explicitly, much of it can be handled since some control information is declaratively embedded in the knowledge base.

The Representation Scheme

The explanation system makes use of the different features of the representation scheme. Most of the important information which one would like to get from the knowledge base can be obtained from the organization, either directly from the different relationships, or by inference.

The representation of knowledge uses concepts from semantic network theory [2] and particularly the PSN formalism [6]. The basic entities of the representation are *frames* which are used to model abstract concepts. The internal structure of a frame is defined by *slots* that form its parts, each slot having a specific *type*, constraints, defaults, relations to other slots, and exceptions. Any slot constraint or relationship may have an associated "at time..." clause specifying the time instant or interval at which the expression is to be evaluated [14]. The exceptions are frames themselves and have slots that are filled with the matching failure characteristics, so that proper selection can be made of alternate frames. This defines an implicit *PART_OF* hierarchy of description. The PART_OF relationship relates a frame to its parts, which are the types of its slots. Slots are classified into two categories: *prerequisite* slots specify concepts which must be observed before the frame can be instantiated, while *dependents* slots provide additional semantic components that are included along with the frame concept on instantiation. Frames are organized along the *IS_A* relationship in order to relate general to specific frames. This provides a mechanism for imposing more global constraints on specific concepts, by means of inheritance of properties from father to son nodes in the hierarchy. In order to assist the motion recognition process, one or more *similarity links* [7] are included with each frame. These links relate frames that are competitors and are used to handle situations where one or more exceptions have been raised. Exceptions are handled via similarity links of the PART_OF parent frame of the failing frame, which provides a context for the exception. An exception raised by a part implies that the frame itself has also failed, since PART_OF is a form of existential quantification. Finally, temporal connectivity is included in the representation scheme in several ways. Each frame has a special slot ("time_int") that defines its temporal boundaries and is itself a frame with 3 slots : start time, end time, and duration. Temporal constraints on each of these qualities can be attached to any slot, providing points of time or intervals with which to represent temporal information, as opposed to interval based operations only [1]. In addition, the PART_OF hierarchy offers event grouping in time.

Classes of Knowledge Queries

As mentioned before, the questions evolve from the structure of the knowledge base. We therefore distinguish between two main aspects of the explanation capability in the system: explaining the contents of specific frames, and explaining the organization of frames.

One important note is the fact that it is expected that the system will be used by different users. Therefore, there will be some variations to questions, depending on the users' background or knowledge of the system. All the examples in this section will be of "canonical" questions, i.e., questions in their extended form, without complex syntax or abbreviated wordings. For some examples of variations in wordings, see [4].

Presenting contents of frames

The system has the ability to extract portions of information found within frames, as requested by the user. Exact definitions of each frame can be presented to the user, revealing the actual quantitative and qualitative data. To illustrate what kind of information is obtained, we present an example of explanation concerning contents of frames.

```
>>> show me the constraints on slot SUBJ in N_SYSTOLE


      subj : N_LV such that [
     find narrow : NARROW where [
       narrow.subj = self.subj ,
       narrow.time_int during self.time_int,
       narrow.speed < 150  and narrow.speed > 50 /* 1 */
      ]
      exception [NARROWING_IMPROPERLY]
     ];
```

Note that in the above example, there is a number between two asterisks on the right hand side (i.e., " /* 1 */ "). This points the user to a certain reference (from the medical literature) which was used by the knowledge engineer when the knowledge base was constructed. This reference can be queried by the user if desired. It is also clear that such a response is useful only to a sophisticated user.

Presenting frame organization

The second set of questions are "organization" questions. The ability to traverse the different organizations allows the user to obtain a general picture of the knowledge base, without getting into detailed descriptions (the contents of frames). Therefore one can see how certain concepts are specialized, how complex concept aggregations are broken down into simpler ones, how the failure of one hypothesis leads to the consideration of a different one, and how events are ordered in time. This constitutes the four dimensions along which one can trace information: the IS_A relationship, the PART_OF relationship, similarity links, and temporal connections. Some examples of organization questions are:

- Show me the IS_A hierarchy rooted at frame F.
- What are the direct PART_OF descendents of frame F?
- Show me the similarity relationships that event E has.
- What event precedes event E?

The questions that involve the IS_A and PART_OF relationships involve traversal of directed graphs (the IS_A and PART_OF hierarchies). This information is presented using a graphics package since it is more convenient to see it in pictorial form. All frames have both IS_A and PART_OF relationships with other frames. Thus, the level of specificity of detail can be controlled by, or examined by traversing, the IS_A hierarchy, while the level of resolution of detail (decomposition) is reflected in the PART_OF hierarchy. An example of an IS_A question can be found in Figure 1.

```
>>> show me the is-a hierarchy rooted at MOTION
```

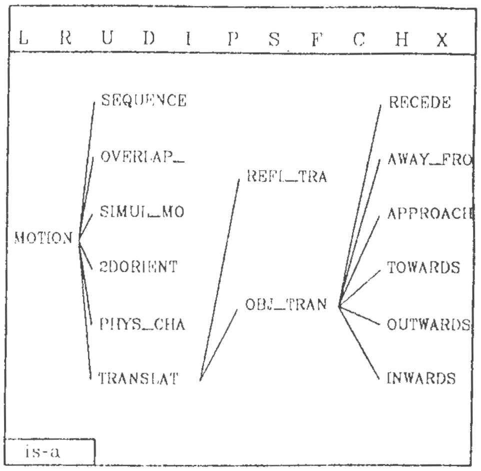
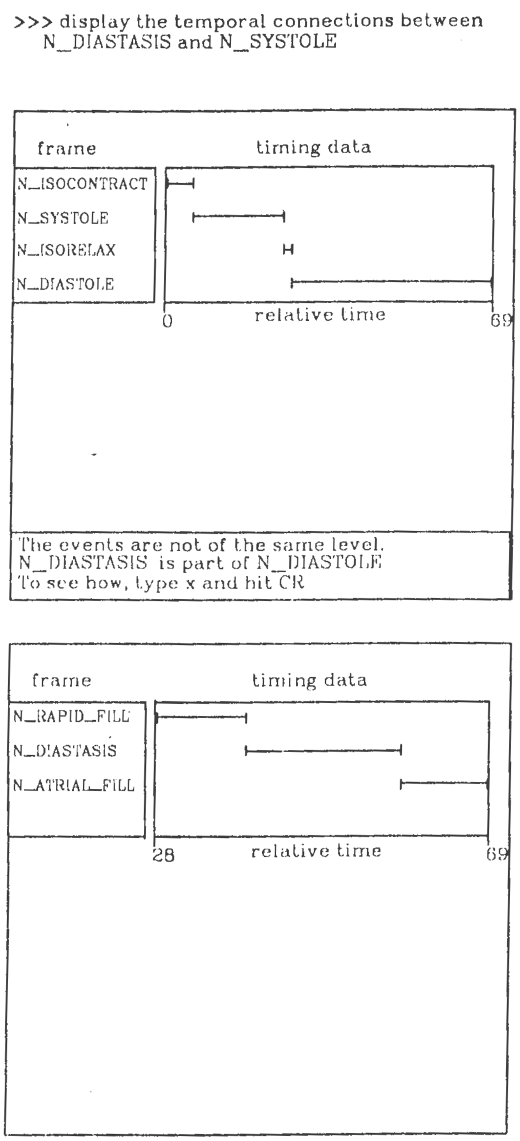


Figure 1

Frame Comparisons

Similarity links connect concepts which have both some common features and some differences, and are considered as competitors in a recognition process. For further details on the control scheme in ALVEN see [12, 14]. If the user asks about frame comparisons, the explanation module must show how the frames differ and how they are similar. The significance of this comparison is the fact that the similarity links are used by the control scheme. Being able to see how and why control is transferred from one hypothesis (frame) to another allows the user to understand what options the system has in case certain hypotheses fail

Initially the system has to find out whether the frames in question are comparable, i.e., check if they are connected via similarity links. There does not have to be a direct connection -- there could be a chain (or chains) of links connecting two frames. Common features are found between chains of frames and are presented as they appear in the "with similarities" component; differences can be compared along a chain of frames, but comparison is conducted only if the same exceptions are involved. The reason is that these exceptions are the driving force that causes one frame to activate another, and following an exception along the similarity chain is a way of examining how one phenomenon can cause the activation of different alternatives. Consequently, if there exists a similarity path of frames with no common exceptions, the result of such a search will be regarded as failure -- no common thread of differences. However, if the system does find a path with common exception types, the constraints attached to the exceptions are examined. If they do not involve the same slot fillers (subjects, referents, etc.) -- the system notes the fact that the exception types themselves were common but not their tokens. On the other hand, if the constraints do involve common slot fillers, the system finds which ones form the difference chain. We present below an example of finding differences between frames.



```
>>> display the temporal connections between
    N_DIASTASIS and N_SYSTOLE
```

```
The events are not of the same level.
N_DIASTASIS is part of N_DIASTOLE
To see how, type x and hit CR
```

Remarks
The upper figure presents the temporal ordering between the higher level events, while the figure below provides more details about the lower level event's place within its ancestor's context.

Figure 3

Implementation

The general design of the explanation system consists of three basic steps -- an input module, the explanation generator, and an output module. The purpose of the input module is to enable the user to ask questions in natural language, with some restrictions, and to activate the corresponding explanation function within the generator. Therefore, a parser was designed and implemented, enabling the system to cope with questions asked in free format and different variations. The output module presents the sought information in a suitable form, either using textual or graphic representation. The textual response is either the contents of frames as they appear in the knowledge base, or lists of frames which share common features as requested by the user, along with some additional remarks. The graphical representation includes tools for presenting hierarchies or parts of them, as well as displaying events on a time scale after they have been temporally ordered by the system.

The parser utilizes a semantic grammar, represented by means of Augmented Transition Networks (ATNs), and an ATN interpreter. Parsing of queries is preceded by a lexical analysis phase during which dictionary look-up is performed and the input is scanned for keywords in order to determine the order of application of the ATN subnets. The grammar has been designed to accommodate variations in the tense, voice, and number features used in the expression of queries by the user. Ellipsis can be employed to almost any extent desirable. There is also some provision to handle so-called "semantic ellipsis": for instance, if a user requests some information involving a particular frame, but fails to specify the identity of the frame, the system fills in the missing information by "guessing" which frame was meant, using a heuristically-guided search backwards through a context list containing the names of entities mentioned in previous queries. The same mechanism is used to resolve pronoun reference. The system informs the user of any substitutions or insertions after the successful completion of the parsing phase for a query. Recognition of a query results in a direct call to the particular explanation module designed to respond to it, followed by an opportunity for the user to continue the dialogue by entering another question.

Summary

The explanation capability described in this paper provides an interactive mechanism for user communication with the knowledge base. It supplies tools with which the user can be exposed to the knowledge base and gather, study, or query its information. It was built upon a knowledge base in which each concept was represented as a frame, and all frames were linked through different relationships. These relationships enable the system to cope with issues such as time, space, and events, whereas other systems, in particular, the rule-based ones, do not offer adequate solutions to these complex issues. Moreover, the explanation of the knowledge base need not end with queries whose answers involve retrieval of chains of rules, but rather handle a variety of cases, such as abstraction along one of several directions (e.g., IS_A, PART_OF), temporal ordering of events, comparisons between different frames, and the ability to extract frames and parts of them. Since the explanation system makes use of the underlying representation, and generates the explanations without using domain specific constructs, it can handle any knowledge base organized similarly. Another important characteristic of this system is the fact that it combines text and diagrams as appropriate in the explanation process.

```
>>> What are the differences between HCM_LV_CYCLE and
                        CCM_LV_CYCLE ?


TOO_HIGH_ESV
----------------------


MINAXIS_TOO_LONG
----------------------

     d3.time_int = systole.time_int.et
     d2.time_int = diastole.time_int.et
```

Remarks:
Here, two differences are observed : TOO_HIGH_ESV (too high end systolic volume), which has no additional constraints attached to it, and MINAXIS_TOO_SHORT (the minor axis is too short), and the constraints are as shown (at the times of end-systole and at end-diastole).

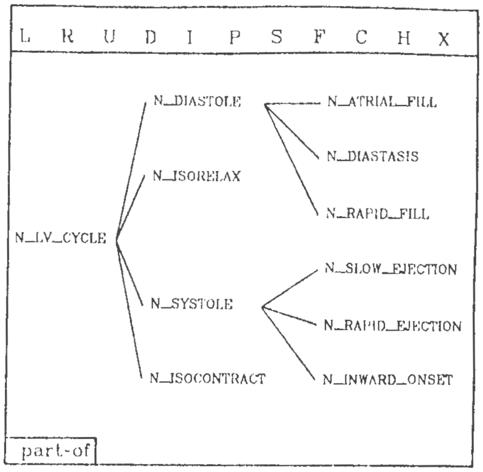


Figure 2

Explanation of Temporal Information

Perhaps the most challenging aspect is the explanation of temporal information. The reason is that some concepts inherit their temporal properties from the IS_A hierarchy, while other temporal information is found within slots and within "at" clauses. All frames in our system which are motion frames (i.e., not object frames) are events. The task of the system is to take these frames as they appear within the knowledge base and map them onto a time scale according to their relative occurrence.

The ALVEN system contains a general motion knowledge base, as well as a domain specific motion knowledge base (left ventricular motion concepts), the domain knowledge being defined in terms of the general knowledge. General motion concepts can be thought of as *temporally unbound motions* while domain specific motion concepts as *temporally bound motions*. In other words, we usually place quantitative temporal restrictions on the motions in the application domain, while general motion concepts remain always unbound quantitatively. Additionally, the design of the explanation capability relies on the fact that there are frames such as SEQUENCE, SIMUL_MOT, and OVERLAP_MOT which exist and define temporal grouping of sets of events.

Temporal Ordering

For the purpose of relating events to each other, the notation suggested in [1] is used. The process of ordering is facilitated by the fact that frames contain all the information (including the temporal ordering) about their immediate descendents. Figure 2 illustrates this point. In N_LV_CYCLE we can learn that N_ISORELAX is met by N_SYSTOLE , so when the traversal reaches the next level, the system already knows that N_SLOW_EJECTION must be met by N_ISORELAX. Therefore, information is local to the frame itself, but it is global to all the components of that frame.

In order to demonstrate the method by which the system orders frames, we shall use one of the questions mentioned before. The problem will be to show the temporal connections between N_DIASTASIS and N_SYSTOLE, and they appear in the PART_OF hierarchy presented in Figure 2.

The first problem the system has to tackle is the fact that the two events are not in the same PART_OF level. It is semantically more appropriate to relate the event of the higher level (in our example -- N_SYSTOLE) to the other one's PART_OF ancestor that shares the same level in the hierarchy (N_DIASTOLE). The lower level event (N_DIASTASIS) appears within its ancestor's (N_DIASTOLE) context. Since the PART_OF hierarchy enables the system's designer to break complex concepts into simpler ones, what this means is that the comparison will be performed between concepts of the same complexity, and then, further information will be provided to show where and how the simpler concept appears within the more complex context. An example of the system's response to the original question can be found in Figure 3.

The process of ordering is done in two phases: (a) the system has to search up the PART_OF hierarchy for a common ancestor to the two events in question, which will be an event containing them or their ancestors as parts; (b) from the common event, the system traverses down the hierarchy, level after level, relating the parts according to their temporal ordering. (The detailed algorithm appears in [5]). The search up the PART_OF hierarchy can be complicated by the fact that one of the events in question, or its ancestors, may have more than one ancestor. This can happen when an event appears in two (or more) different contexts, in which case the system asks the user to resolve the ambiguity. For the simplicity of our example, Figure 2 is presented as a tree, i.e., all the ambiguities have already been resolved by the user.

Acknowledgements

We wish to thank John Tsotsos for his encouragement and his helpful suggestions. Ray Perrault provided helpful advice on the design of the natural language interface. Thanks are also expressed to Andrew Gullen and to Brian Nixon for their comments and advice.

This research has been supported in part by the Canadian Heart Foundation, the Ontario Heart Foundation, and by the Defence Research Establishment Atlantic.

References

[1] Allen, J.F. "Maintaining knowledge about temporal intervals", University of Rochester - Dept. of Computer Science TR-86, 1981

[2] Brachman, R.J. "On the epistemological status of semantic networks", in *Associative networks: Representation and use of knowledge by computers* (Findler, N.V., ed.). Academic Press, New York, 1979

[3] Davis, R. "Interactive transfer of expertise: Acquisition of new inference rules", *Artificial Intelligence 12,* 1979

[4] Gershon, R., Ali, Y., and Jenkin, M. "An explanation system for frame-based knowledge organized along multiple dimensions", LCM-TR83-2, Laboratory Computational Medicine, Dept. of Computer Science, University of Toronto, 1983

[5] Gershon, R. "Explanation methods for visual motion understanding systems", M.Sc. Thesis, Dept. of Computer Science, University of Toronto, 1982

[6] Levesque, H., and Mylopoulos, J. "A procedural semantics for semantic network", in *Associative networks: Representation and use of knowledge by computers* (Findler, N.V., ed.), Academic Press, New York, 1979

[7] Minsky, M. "A framework for representing knowledge", in *The Psychology of Computer Vision,* (Winston, P., ed.), McGraw-Hill, 1975

[8] Pauker, S.G., Gorry, G.A., Kassirer, J.P., and Schwartz, W.B. "Toward the simulation of clinical cognition: Taking present illness by computer", *The American Journal of Medicine 60,* 1976

[9] Pople, H.E. "The formation of composite hypothesis in diagnostic problem solving: An exercise in synthetic reasoning", *Proc. of the Fifth International Joint Conference on Artificial Intelligence,* 1977

[10] Shortliffe, E.H. *Computer based medical consultation: MYCIN.* Elsevier North Holland Inc., 1976

[11] Swartout, W.R. "Producing explanations and justifications of expert consulting programs", M.I.T. Laboratory for Computer Science TR-251, 1981

[12] Tsotsos, J.K. "Representation axes and temporal cooperative processes", in *Vision, Brain and Cooperative Computation,* Arbib, M., and Hanson, A. (eds.), in press

[13] Tsotsos, J.K. "Temporal event recognition: An application to left ventricle performance", *Proc. of the Seventh International Joint Conference on Artificial Intelligence,* 1981

[14] Tsotsos, J.K. "A framework for visual motion understanding", Ph.D. dissertation, Technical Report CSRG-114, Dept. of Computer Science -- University of Toronto, 1980

[15] van Melle, W.J. "A domain independent system that aids in constructing knowledge based consultation programs", Ph.D. Dissertation, Stanford University -- Dept. of Computer Science, STAN-CS-80-820, 1980

QUALITATIVE SENSITIVITY ANALYSIS: AN APPLICATION TO EXPERT SYSTEM PLAN JUSTIFICATION

Stephen E. Cross[1]
Artificial Intelligence Laboratory
Department of Electrical Engineering
Air Force Institute of Technology
Wright-Patterson AFB, Ohio 45433

Abstract

A working computer program, an air traffic control expert system, generates knowledge that human air traffic controllers use to justify plans. The knowledge is derived from mathematical knowledge of aircraft performance by qualitative simulation. The computer translates the equations into a semantic structure which provides a framework for qualitative simulation. Simulation results are interpreted at an abstract level that represents the human controller's understanding of naive physics. The process is called *qualitative sensitivity analysis*. The approach is unique in three aspects. First, an abstraction level is used. The equations can be interpreted in terms of Newtonian mechanics as applied to the one dimensional motion. Second, the reasoning process is bidirectional. Third, the computer generates the semantic structure based on a symbolic series expansion of the equations.

1.0. Introduction

Pilots often 'talk with their hands.' During a mission prebriefing, a fighter pilot will use hand gestures to indicate how individual aircraft are to approach the lead aircraft, fly in formation, and accomplish other mission related goals. In essence, the pilot is justifying plans for specific mission goals using sensitivity analysis based on a naive understanding of the aircraft equations of motion. This type of reasoning is common to many human problem solving domains. For instance, the driver of an automobile can justify why one would decrease gears while climbing a hill. Air traffic controllers justify their plans and the plans of other controllers (including novel plans which may be more elegant than their own) through a similar reasoning process. A common aspect is that the physics of the machine or machines being controlled by the human can be modeled with mathematical equations and that humans produce and understand justifications based on their naive physics understanding of the equations [1]. The reasoning process, qualitative sensitivity analysis, enables the computer to acquire knowledge in a form useful for plan justification.

The approach is unique in three aspects. First, a level of abstraction is included. Domain equations may be computationally too complex for a human expert to use. However, the equations can be interpreted in terms of a naive representation of Newton's laws as applied to one dimensional motion thus abstracting the influences inherent in the equations. Second, the approach enables bidirectional reasoning. Qualitative knowledge can be used to direct quantitative reasoning. Additionally, when new equations are implemented, their meaning is represented explicitly and interpreted using the existing qualitative knowledge. Third, the computer constructs its own representation of the equations based on a symbolic series expansion.

2.0. An Example Problem

An illustrative problem from air traffic control is used in the remaining sections of the paper. Consider two aircraft that are involved in a 'head-on' conflict. The controller must generate a plan that prevents a mid-air collision. The plan must involve the modification of one of the aircraft's flight plans. If collision avoidance were the only air traffic control goal, the solution would be trivial. Any legal operator (.e.g., climb, descend) could be used, but there are other important goals such as fuel efficiency. A significant portion of the controller's training involves assimilating heuristics useful for generating plans that achieve both goals. For instance, aircraft are usually more fuel efficient at higher altitudes. The human controller choses an operator that prevents a collision and improves (or at least not seriously degrades) fuel efficiency. The strategies and tactics one controller uses may vary from those of another controller. However, a human controller can easily understand the plans of another controller. The justifications use knowledge of naive physics and the aircraft equations of motion.

Consider two aircraft in a head-on conflict and assume both aircraft have an enroute goal (fly a fuel efficient path cross country). The expert system solves this problem by using heuristics about aircraft intentions and the conflict goals to cause one of the aircraft to climb to a higher altitude. The justification is based on the knowledge that cross country aircraft are more fuel efficient at higher altitudes because as they burn off fuel, they become lighter and thus encounter less resistive force at a higher altitude. The same solution was obtained in a scenario when of the cross country aircraft was a recently inflight-refueled military transport aircraft. When the computer was advised to descend the military aircraft, the computer was able to generate the knowledge that since the military aircraft's mass increased (due to the inflight refueling) it was more fuel efficient at a lower altitude. The reasoning process that allows this type of knowledge to be translated from mathematical models is now discussed.

3.0. The Reasoning Component

The reasoning component consists of naive physics knowledge, domain equations, and interfaces between these knowledge types and a data base of heuristics. The reasoning component serves the role of a knowledge translator. That is, the reasoning component can generate heuristic-like knowledge from a mathematical substructure.

3.1. Naive Physics Level

Newton's laws are represented in a semantic network where nodes are primitive concepts (e.g, force, velocity) and links indicate their interaction. The structure represents the author's understanding of Newtonian mechanics as abstracted to one dimensional motion of mechanical systems. The naive physics knowledge is made explicit by the designer. The linkages between the naive physics level and a domain are equation dependent. The representation serves as the basis for the interpretation of yet-to-be specified equations and algorithms.

[1] This research was accomplished at the University of Illinois and was supported by the Department of Transportation, Federal Aviation Administration under contract number DOT-FA79WA-4360.

Nodes represent forces (propulsive, enabling, resistive), acceleration, velocity, position, mass, and power. Propulsive and enabling forces are referred to as positive forces. Propulsive forces require an external enablement which converts fuel (portion of system's mass) into the force. The rate of change of mass varies directly with the change in propulsive force. That is, when propulsive force increases, the fuel flow rate increases causing the mass rate of change to increase.

Another positive force is called an enabling force. Enabling forces do not directly influence mass. For instance, the air flow over a wing causes a pressure differential that enables lift. Lift is an enabling force that counteracts weight. Resistive forces counteract positive forces. Common examples are pressure and friction forces which vary with velocity and domain dependent variables such as weight, density, and surface area.

Nodes are related by links which specify how an independent variable is 'influenced' by a changing dependent variable. The links define a structure in which qualitative values are propagated. There are four types of links: influences, component, parent, and instance. Influence links are labeled with two attributes: primary/secondary and positive/negative. For example, acceleration is primarily influenced by force and secondarily influenced by mass. The primary/secondary attribute is used to identify variables that are normally static thus improving the search efficiency. The positive/negative attribute indicates the direction of change of a variable. For instance, mass is a negative influence on acceleration. When mass increases, acceleration decreases (assuming that force does not changes and the body is in motion). Component links are used to partition equations into semantically relevant subterms. For instance, aircraft drag has two components: parasitic drag and induced drag. Parent and instance links indicate hierarchical relationships (often identified as an 'ISA' link).

3.2. Domain Equations

Aircraft equations of motion are dependent on four forces: lift (L), thrust (T), drag (D), and weight (W). For level flight, dynamic equilibrium is defined as:

$$L - W = 0 \tag{1}$$

$$T - D = 0 \tag{2}$$

Each force is defined by an equation. Thrust is a function of throttle setting. Lift is a function of velocity, air density, angle of attack, and wing geometry. Weight is a function of aircraft mass and gravity. Drag is a function of air density, velocity, and aircraft configuration variables. Drag has two components: parasitic and induced drag. All subsonic aircraft performance capabilities can be derived from the drag equation (3). Consider an aircraft at a constant altitude and configuration. The maximum velocity then occurs when the drag equals the maximum thrust. Since drag is parabolic with velocity, the velocity at the minimum drag defines the 'best endurance' airspeed~

$$D = D_p + D_i \tag{3}$$

$$D_p = \frac{f\ V^2\ \sigma}{295} \tag{4}$$

$$D_i = \frac{94}{\sigma e}\left(\frac{w}{b}\right)^2 V^{-2} \tag{5}$$

where,

v = velocity
w = weight
σ = altitude density ratio
b, e, and f = aircraft configuration variables

A symbolic series expansion is used to define the influence links. The sign of the first error term indicates a positive or negative influence. The magnitude of the influence is the amount of the first error term and is saved if ambiguity resolution is later required. A variable is defined as a primary influence if it is an instance of an abstracted concept and that concept is also a primary influence. Aircraft airspeed (or horizontal velocity) is a primary influence because it is an instance of velocity which is itself a primary influence of position. Sometimes influences are found recursively. For instance, air density is a function of altitude (an instance of vertical position). Position can be a primary influence of resistive force (e.g., friction). Thus, it is inferred that air density is a primary influence of drag.

3.3. Interfaces

The sign of the terms in the force equations in (1) and (2) are used to define instances of forces in the naive physics representation. For instance, T and L are positive forces. Semantic knowledge is also required. Thrust is a propulsive force while lift is an enabling force. In this context, weight and drag are resistive forces. The representation of both aircraft levels and the abstracted structure is shown in Fig. 1. Conceptual knowledge specifies the procedures to use in different contexts. A plan is fuel efficient if after implementing the plan, fuel consumption decreases. Less fuel is used if the drag decreases since thrust equals drag in level flight. Consider the case where a climb command is given. From the structure of Fig. 1, it is seen that the air density term decreases. This may or may not cause a favorable change in drag. The effect of error terms on the drag components due to changes in density are retrieved and the pertinent computation is performed.

4.0. Artificial Intelligence Issues

Hayes [2] discussed the need for naive theories and provided a theoretical framework for future work. de Kleer [3] explored the computational aspects of qualitative reasoning in the mini-world of the roller coaster and contributed the concept of envisionment. Envisionment predicts system behavior through qualitative simulation. In related research [4], he illustrates the use of Incremental Qualitative (IQ) analysis as a weak form of reasoning about perturbation. Forbus [5] uses a stronger approach that includes the sign and magnitude of a quantity's amount and derivative. In essence, Forbus enables the computer to perform a sensitivity analysis.

Qualitative sensitivity analysis generates the knowledge that humans sometimes use for plan justification. The traditional approach to plan justification is based on non-monotonic dependency networks and associated processes (e.g., data-dependent backtracking) [6,7] and requires that the knowledge be represented in a uniform and convenient form. Knowledge may reside at many levels and in many forms. A data-dependent backtracking scheme for plan justification would first need to acquire and represent the relevant knowledge. In this work, the computer can acquire the relevant knowledge.

The justification for the climb command is that it prevents a mid-air collision and is fuel efficient. The important point is that the computer understands the fuel efficient aspect in terms of the equation and its semantic structure. It can explain its justification at a more abstract level. The decrease in air density causes drag to decrease. Drag is a resistive force. Since resistive force decreases, the positive forces can also decrease. Similarly, a novel plan can be justified. An aircraft that increased its mass was found to be more fuel efficient at a lower altitude by qualitatively simulating the influences of the increased mass.

Propagation of constraints in the overall structure provides a form of common-sense reasoning about the represented domain. A controller should never issue a command that cannot be implemented by the aircraft. For instance, consider that an aircraft is commanded to increase its velocity. Implicit in the command is that the aircraft is to maintain its present altitude. The computer can perform a qualitative simulation of the command subject to constraints. The computer finds that (1) the

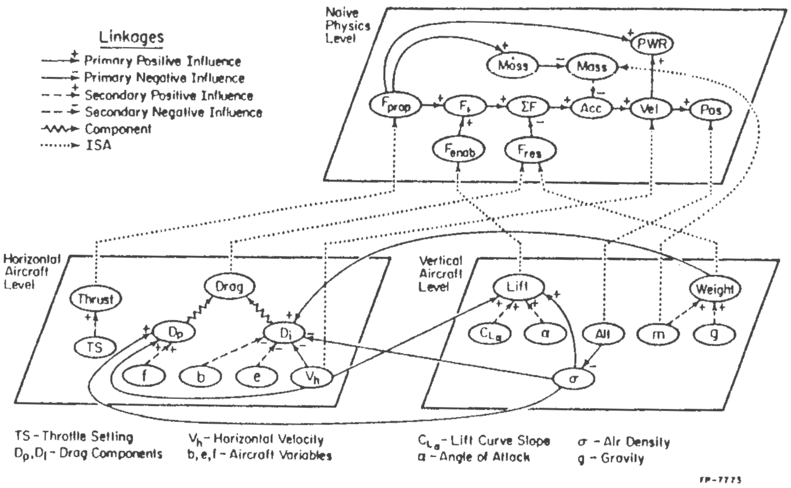


Figure 1: Interface to the Aircraft Spaces

aircraft increases its speed by increasing its throttle setting and (2) it maintains its present altitude by decreasing its angle of attack. The answers were obtained by a combination of forward and backward constraint propagation.

The approach may be useful in learning new problem solving strategies. Human controllers acquire their skill by the assimilation and justification of an expert controller's plans. The air traffic control expert system [1] is a script-based and DeJong's Explanatory Schema Acquisition theory [8] is being investigated as a potential learning module. An important aspect of building new schemata is obtaining the correct knowledge. Recent research has stressed the importance of understanding the knowledge and translating the knowledge into a useful form [9,10]. The computer can transform mathematical knowledge into a form useful for novel plan justification in the domain of air traffic control (as well as the automobile domain in a limited test). Research has begun on how to use the knowledge for automated plan justification and how to represent the knowledge in schemata.

6.0. Conclusions

Qualitative sensitivity analysis has been used by an air traffic control expert system to facilitate the integration of mathematical and heuristic knowledge. The reasoning is accomplished using a computer generated semantic network which represents the domain equations. The space of domain equations is referenced to an abstract space that represents Newton's laws as applied to one dimensional motion. A qualitative simulation based on constraint propagation was useful for generating the type of naive physics knowledge that human controllers use in plan justification. The future research goals are to represent this knowledge in a heuristic form (i.e., to learn) and to use the knowledge for plan justification.

References

[1] Cross, S. "An Approach to Plan Justification Using Sensitivity Analysis," to appear in *IEEE Transactions on Systems, Man, and Cybernetics.*

[2] Hayes, P. 'The Naive Physics Manifesto," in *Expert Systems in the Micro-Electronic Age*, D. Michie (Ed.), Edinburgh: Edinburgh University Press, 1979.

[3] de Kleer, J. 'Qualitative and Quantitative Reasoning in Classical Mechanics," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*. pp. 229-304, 1977.

[4] de Kleer, J. *Causal and Teleological Reasoning in Circuit Recognition*, Ph.D. Dissertation, Massachusetts Institute of Technology, MIT-AI-TR-529, 1979,

[5] Forbus, K. *Qualitative Process Theory*, Massachusetts Institute of Technology, MIT-AI-TR-664, 1982.

[6] Doyle, J. "A Truth Maintenance System," *Artificial Intelligence*, 12:231-272, 1979

[7] O'Rorke, P. 'Reasons for Beliefs in Understanding: Applications of Non-monotonic Dependencies to Story Understanding," Proceedings of the National Conference on Artificial Intelligence, Washington, D.C., pp. 306-309, 1983.

[8] DeJong, G. "Acquiring Schemata through Understanding and Generating Plans," Proceedings of the Eight International Joint Conference on Artificial Intelligence, Karlsrule, West Germany, pp. 462-464, 1983.

[9] Mostow, D. *Mechanical Transformation of Task Heuristics into Operational Procedures*, Ph. D. Dissertation, Carnegie-Mellon University, 1981.

[10] Keller, R. 'Learning By Re-expressing Concepts for Efficient Recognition," *Proceedings of the National Conference on Artificial Intelligence, Washington, D.C., pp. 182-186, 1983.*

A Fundamental Tradeoff in Knowledge Representation and Reasoning

Hector J. Levesque

Fairchild Laboratory for AI Research
Palo Alto, CA

This paper examines from a general point of view a basic computational limit on automated reasoning, and the effect that it has on Knowledge Representation (KR). The problem is essentially that it can be more difficult to reason correctly with one representational language than with another and, moreover, that this difficulty increases as the expressive power of the language increases. So there is a tradeoff between the expressiveness of a representational language and its computational tractability. What we attempt to show is that this tradeoff underlies the differences among a number of representational formalisms (such as first-order logic, databases, semantic networks, frames) and motivates many current research issues in KR (such as the role of analogues, syntactic encodings, and defaults, as well as the systems of limited inference and hybrid reasoning).

To deal with a such a broad range of representational phenomena, we must, of necessity, take a considerably simplified and incomplete view of KR. In particular, we focus on its computational and logical aspects, more or less ignoring its history and relevance in the areas of psychology, linguistics, and philosophy. The area of KR is still very disconnected today and the role of logic remains quite controversial, despite what this paper might suggest. We do believe, however, that the tradeoff discussed here is fundamental. As long as we are dealing with computational systems that reason automatically (without any special intervention or advice) and correctly (once we define what *that* means), we will be able to locate where they stand on the tradeoff: they will either be limited in what knowledge they can represent or unlimited in the reasoning effort they might require.

Our computational focus will not lead us to investigate specific algorithms and data structures for KR and reasoning, however. What we discuss is something much stronger, namely whether or not algorithms of a certain kind can exist at all. So the analysis here is at the *Knowledge Level* [1] where we look at the content of what is represented (in terms of what it says about the world) and not the symbolic structures used to represent that knowledge. Indeed, we examine specific representation schemes in terms of what knowledge they can or cannot represent, rather than in terms of how they might actually represent it.

In the first section below, we discuss what a KR system is for and what it could mean to reason correctly. Next, we investigate how a KR service might be realized using theorem proving in first-order logic and the problem this raises. Following this, we present various representational formalisms and examine the special kinds of reasoning they suggest. Finally, we draw some tentative conclusions from this analysis.

1. The Role of KR

While it is generally agreed that KR plays an important role in (what have come to be called) knowledge-based systems, the exact nature of that role is often hard to define. In some cases, the KR subsystem does no more than manage a collection of data structures, providing, for example, suitable search facilities; in others, the KR subsystem is not really distinguished from the rest of the system at all and does just about everything: make decisions, prove theorems, solve problems, and so on. In this section, we discuss in very general terms the role of a KR subsystem within a knowledge-based system.

1.1. The KR Hypothesis

A good place to begin a discussion of KR as a whole is with what Brian Smith has called in [2] the *Knowledge Representation Hypothesis:*

> *Any mechanically embodied intelligent process will be comprised of structural ingredients that a) we as external observers naturally take to represent a propositional account of the knowledge that the overall pro-*

> *cess exhibits, and b) independent of such external semantical attribution, play a formal but causal and essential role in engendering the behaviour that manifests that knowledge.*

This hypothesis seems to underly much of the research in KR. In fact, we might think of *knowledge-based systems* as those that satisfy the hypothesis by design. Also, in some sense, it is only with respect to this hypothesis that KR research can be distinguished from any number of other areas involving symbolic structures such as database management, programming languages and data structures.

Granting this hypothesis, there are two major properties that the structures in a knowledge-based system have to satisfy. First of all, it must be possible to interpret them as *propositions* representing the overall knowledge of the system. Otherwise, the representation would not necessarily be of *knowledge* at all, but of something quite different, like numbers or circuits. Implicit in this constraint is that the structures have to be expressions in a language that has a *truth theory.* We should be able to point to one of them and say what the world would have to be like for it to be true. The structures themselves need not *look* like sentences—there are no syntactic requirements on them at all, other than perhaps finiteness—but we have to be able to understand them that way.

A second requirement of the hypothesis is perhaps more obvious. The symbolic structures within a knowledge-based system must play a *causal role* in the behaviour of that system, as opposed to, say, comments in a programming language. Moreover, the influence they have on the behaviour of the system should agree with our understanding of them as propositions representing knowledge. Not that the system has to be aware in any mysterious way of the interpretation of its structures and their connection to the world;[1] but for us to call it knowledge-based, *we* have to be able to understand its behaviour as if it believed these propositions, just as we understand the behaviour of a numerical program as if it appreciated the connection between bit patterns and abstract numerical quantities.

1.2. Knowledge Bases

To make the above discussion a bit less abstract, we can consider a very simple task and consider what a system facing this task would have to be like for us to call it knowledge-based. The amount of knowledge the system will be dealing with will, of course, be very small.

Suppose we want a system in PROLOG that is able to print the colours of various items. One way to implement that system would be as follows:

```
printColour(snow) :-
    !, write("It's white.").
printColour(grass) :-
    !, write("It's green.").
printColour(sky) :-
    !, write("It's yellow.").
printColour(X) :- write("Beats me.").
```

A slightly different organization that leads to the same overall behaviour is

```
printColour(X) :-
    colour(X,Y), !, write("It's "),
    write(Y), write(".").
printColour(X) :- write("Beats me.").

colour(snow,white).
colour(grass,green).
colour(sky,yellow).
```

The second program is characterized by explicit structures representing the (minimal) knowledge[2] the system has about colours and is the kind of system that we are calling knowledge-based. In the first program, the association between the object (we understand as) referring to grass and the one referring to its colour is implicit in the structure of the program. In the second, we have an explicit *knowledge base* (or KB) that we can understand as propositions relating the items to their colours. Moreover, this interpretation is justified in that these structures determine what the system does when asked to print the colour of a particular item.

One thing to notice about the example is that it is not the use of a certain programming language or data-structuring facility that makes a system knowledge-based. The fact that PROLOG happens to be understandable as a subset of first-order logic is

[1] Indeed, part of what philosophers have called the *formality condition* is that computation at some level has to be uninterpreted symbol manipulation.

[2] Notice that typical of how the term "knowledge" is used in AI, there is no requirement of *truth.* A system may be mistaken about the colour of the sky but still be knowledge-based.

largely irrelevant. We could probably read the first program "declaratively" and get sentences representing some kind of knowledge out of it; but these would be very strange ones dealing with writing strings and printing colours, not with the colours of objects.

1.3. The KR Subsystem

In terms of its overall goals, a knowledge-based system is not directly interested in what specific structures might exist in its KB. Rather, it is concerned about what the application domain is like, for example, what the colour of grass is. How that knowledge is represented and made available to the overall system is a secondary concern and one that we take to be the reponsibility of the KR subsystem. The role of a KR subsystem, then, is to manage a KB for a knowledge-based system and present a picture of the world based on what it has represented in the KB.

If, for simplicity, we restrict our attention to the yes-no questions about the world that a system might be interested in, what is involved here is being able to determine what the KB says regarding the truth of certain sentences. It is not whether the sentence itself is present in the KB that counts, but whether its truth is *implicit* in the KB. Stated differently, what a KR system has to be able to determine, given a sentence α, is the answer to the following question:

> *Assuming the world is such that what is believed is true, is α also true?*

We will let the notation KB $\models \alpha$ mean that α is implied (in this sense) by what is in the KB.

One thing to notice about this view of a KR system is that the service it provides to a knowledge-based system depends only on the truth theory of the language of representation. Depending on the particular truth theory, determining if KB $\models \alpha$ might require not just simple retrieval capabilities, but also *inference* of some sort. This is not to say that the *only* service to be performed by a KR subsystem is question-answering. If we imagine the overall system existing over a period of time, then we will also want it to be able to augment the KB as it acquires new information about the world.[3] In other words, the responsibility of the KR system is to select appropriate *symbolic structures* to represent knowledge, and to select appropriate *reasoning mechanisms* both to answer questions and to assimilate new information, in accordance with the truth theory of the underlying representation language.

So our view of KR makes it depend only on the semantics of the representation language, unlike other possible accounts that might have it defined in terms of a set of formal symbol manipulation routines (e.g., a proof theory). This is in keeping with what we have called elsewhere a *functional* view of KR (see [3] and [4]), where the service performed by a KR system is defined separately from the techniques a system might use to realize that service.

2. The Logical Approach

To make a lot of the above more concrete, it is useful to look at an example of the kinds of knowledge that might be available in a given domain and how it might be represented in a KB. The language that will be used to represent knowledge is that of a standard first-order logic (FOL).

2.1. Using First-Order Logic

The first and most prevalent type of knowledge to consider representing is what might be called simple *facts* about the world, such as

- Joe is married to Sue.
- Bill has a brother with no children.
- Henry's friends are Bill's cousins.

These might be complicated in any number of ways, for example, by including time parameters and certainty factors.

Simple observations such as these do not exhaust what might be known about the domain, however. We may also have knowledge about the *terminology* used in these observations, such as

- *Ancestor* is the transitive closure of *parent*.
- *Brother* is *sibling* restricted to males.
- *Favourite-cousin* is a special type of *cousin*.

These could be called definitions except for the fact that necessary and sufficient conditions might not always be available (as in the last example above). In this sense, they are much more like standard dictionary entries.

The above two example sets concentrate on what might be called *declarative* knowledge about the world.

[3] It is this management of a KB over time that makes a KR subsystem much more than just the implementation of a static deductive calculus.

We might also have to deal with *procedural* knowledge that focuses not on the individuals and their interrelationships, but on *advice* for reasoning about these. For example, we might know that

- To find the father of someone, it is better to search for a parent and then check if he is male, than to check each male to see if he is a parent.
- To see if x is an ancestor of y, it is better to search up from y than down from x.

One way to think of this last type of knowledge is not necessarily as advice to a reasoner, but as declarative knowledge that deals implicitly with the combinatorics of the domain as a whole.

This is how the above knowledge might be represented in FOL.

1. The first thing to do is to "translate" the simple facts into sentences of FOL. This would lead to sentences like

   $\forall x\,\mathrm{Friend}(\mathrm{henry}, x) \equiv \mathrm{Cousin}(\mathrm{bill}, x)$.

2. To deal with terminology in FOL, the easiest way is to "extensionalize" it, that is, to pretend that it is a simple observation about the domain. For example, the *brother* statement above would become[4]

   $\forall x \forall y\, \mathrm{Brother}(x, y) \equiv (\mathrm{Sibling}(x, y) \wedge \mathrm{Male}(y))$.

3. Typically, the procedural advice would not be represented explicitly at all in an FOL KB, but would show up in the *form* of (1) and (2) above. Another alternative would be to use extra-logical annotations like the kind used in PROLOG or those described in [5].

The end result of this process would be a first-order KB: a collection of sentences in FOL representing what was known about the domain. A major advantage of FOL is that given a yes-no question also expressed in this language, we can give a very precise definition of KB $\models \alpha$ (and thus, under what conditions the question should be answered *yes*, *no*, or *unknown*):

KB $\models \alpha$ iff every interpretation satisfying the sentences in the KB also satisfies α.[5]

There is, moreover, another property of FOL which helps solidify the role of KR. If we assume that the KB is a finite set of sentences and let *KB* stand for their conjunction, it can be shown that

$$\mathrm{KB} \models \alpha \quad \text{iff} \quad \vdash (KB \supset \alpha).$$

In other words, the question as to whether or not the truth of α is implicit in the KB reduces to whether or not a certain sentence is a *theorem* of FOL. Thus, the question-answering operation becomes one of *theorem proving* in FOL.

2.2. The Problem

The good news in reducing the KR service to theorem proving is that we now have a very clear, very specific notion of what the KR system should do; the bad news is that it is also clear that *this service cannot be provided.* The sad fact of the matter is that deciding whether or not a sentence of FOL is a theorem (i.e., the decision problem) is unsolvable. Moreover, even if we restrict the language practically to the point of triviality by eliminating the quantifiers, the decision problem, though now solvable, does not appear to be solvable in anywhere near reasonable time.[6] It is important to realize that this is not a property of particular algorithms that people have looked at but of the *problem* itself: there *cannot* be an algorithm that does the theorem proving correctly in a reasonable amount of time. This bodes poorly, to say the least, for a service that is supposed to be only a part of a larger knowledge-based system.

One aspect of these intractability results that should be mentioned, however, is that they deal with the *worst case* behaviour of algorithms. In practice, a given theorem proving algorithm may work quite well. In other words, it might be the case that for a wide range of questions, the program behaves properly, even though it can be shown that there will always be short questions whose answers will not be returned for a very long time, if at all.

How serious is the problem, then? To a large extent this depends on the kind of question you would like to ask of a KR subsystem. The worst case prospect might be perfectly tolerable if you are interested in a mathematical application and the kind of question you ask is an open problem in mathematics. Provided progress is being made, you might be quite willing

[4] This is a little misleading since it will make the *brother* sentence appear to be no different in kind from the one about Henry's friends, though we surely do not want to say that Henry's friends are *defined* to be Bill's cousins.

[5] The assumption here is that the semantics of FOL specify in the usual way what an interpretation is and under what conditions it will satisfy a sentence.

[6] Technically, the problem is now co-NP-complete, meaning that it is strongly believed to be computationally intractable.

to stop and redirect the theorem prover after a few months if it seems to be thrashing. Never mind worst case behaviour; this might be the *only* case you are interested in.

But imagine, on the other hand, a robot that needs to know about its external world (such as whether or not it is raining outside or where its umbrella is) before it can act. If this robot has to call a KR system utility as a subroutine, the worst case prospect is much more serious. Bogging down on a logically difficult but low-level subgoal and being unable to continue without human intervention is clearly an unreasonable form of behaviour for something aspiring to intelligence.

Not that "on the average" the robot might not do alright. The trouble is that nobody seems to be able to characterize what an "average" case might be like.[7] As responsible computer scientists, we should not be providing a general inferential service if all that we can say about it is that by and large it will probably work satisfactorily. If the KR service is going to be used as a utility and is not available for introspection or control, then it had better be *dependable* both in terms of its correctness and the resources it consumes. Unfortunately, this seems to rule out a service based on theorem proving.

2.3. Two Pseudo-solutions

There are at least two fairly obvious ways to minimize the intractability problem. The first is to push the computational barrier as far back as possible. The area of Automatic Theorem Proving has concentrated on techniques for avoiding redundancies and speeding up certain operations in theorem provers. Significant progress has been achieved here, allowing open questions in mathematics to be answered. Along similar lines, VLSI architectural support stands to improve the performance of theorem provers at least as much as it would any search program.

The second way to make theorem provers more usable is to relax our notion of correctness. A very simple way of doing this is to make a theorem proving program always return an answer after a certain amount of time.[8] If it has been unable to prove either that a sentence or its negation is implicit in the KB, it could assume that it was independent of the KB and answer *unknown* (or maybe reassess the importance of the question and try again). This form of error (i.e., one introduced by an incomplete theorem prover), is not nearly as serious as returning a *yes* for a *no*, and is obviously preferrable to an answer that never arrives. This is of course especially true if the program uses its resources wisely, in conjunction with the first suggestion above.

However, from the point of view of KR, both of these are only pseudo-solutions. Clearly, the first one alone does not help us guarantee anything about an inferential service. The second one, on the other hand, might allow us to guarantee an answer within certain time bounds, but would make it very hard for us to specify what that answer would be. If we think of the KR sevice as reasoning according to a certain logic, then the logic being followed is immensely complicated (compared to that of FOL) when resource limitations are present. Indeed, the whole notion of the KR system calculating what is implicit in the KB (which was our original goal) would have to be replaced by some other notion that went beyond the truth theory of the representation language to include the inferential power of a particular theorem proving program. In a nutshell, we can guarantee getting an answer, but not the one we wanted.

One final observation about this intractability is that it is *not* a problem that is due to the formalization of knowledge in FOL. If we assume that the goal of our KR sevice is to calculate what is implicit in the KB, then as long as the truth theory of our representation language is upward-compatible with that of FOL, we will run into the same problem. In particular, using English (or any other natural or artificial language) as our representation language does not avoid the problem as long as we can express in it at least what FOL allows us to express.

3. Expressiveness and Tractability

It appears that we have run into a serious difficulty in trying to develop a KR service that calculates what is implicit in a KB and yet does so in a reasonable amount of time. One option we have not yet considered, however, is to *limit* what can be in the KB so that its implications are more manageable computationally. Indeed, as we will demonstrate in this section, much of the research in KR can be construed as trading off expressiveness in a representation lan-

[7] This seems to account more than anything for the fact that there are so few average case results regarding decidability.

[8] The resource limitation here should obviously be a function of how important overall it might be to answer the question.

guage for a more tractable form of inference. Moreover, unlike the restricted dialects of FOL analyzed in the logic and computer science literatures (e.g., in terms of nestings of quantifiers), the languages considered here have at least proven themselves quite useful in practice, however contrived they may appear on the surface.

3.1. Incomplete Knowledge

To see where this tradeoff between expressiveness and tractability originates, we have to look at the use of the expressive power of FOL in KR and how it differs from its use in mathematics.

In the study of mathematical foundations, the main use of FOL is in the formalization of infinite collections of entities. So, for example, we have first-order number and set theories that use quantifiers to range over these classes, and conditionals to state what properties these entities have. This is exactly how Frege intended his formalism to be used.

In KR, on the other hand, the domains being characterized are usually finite. The power of FOL is used not so much to deal with infinities, but to deal with *incomplete knowledge* [6]. Consider the kind of facts[9] that might be represented using FOL:

1. ¬Student(john).

   This sentence says that John is not a student without saying what John is.

2. Parent(sue,bill) ∨ Parent(sue,george).

   This sentence says that either Bill or George is a parent of Sue, but does not specify which.

3. $\exists x$ Cousin(bill,x) ∧ Male(x).

   This sentence says that Bill has at least one male cousin but does not say who that cousin is.

4. $\forall x$ Friend(george,x) ⊃ $\exists y$ Child(x, y).

   This sentence says that all of George's friends have children without saying who those friends or their children are or even if there are any.

The main feature of these examples is that FOL is not used to capture complex details about the domain, but to avoid having to represent details that may not be known. *The expressive power of FOL determines not so much what can be said, but what can be left unsaid.*

For a system that has to be able to acquire knowledge in a piecemeal fashion, there may be no alternative to using all of FOL. But if we can restrict the kind of the incompleteness that has to be dealt with, we can also avoid having to use the full expressiveness of FOL. This, in turn, might lead to a more manageable inference procedure.

The last pseudo-solution to the tractability problem, then, is to restrict the logical form of the KB by controlling the incompleteness of the knowledge represented. This is still a pseudo-solution, of course. Indeed, provably, there cannot be a *real* solution to the problem. But this one has the distinct advantage of allowing us to calculate exactly the picture of the world implied by the KB, precisely what a KR service was supposed to do. In what follows, we will show how restricting the logical form of a KB can lead to very specialized forms of inference.

3.2. Database Form

The most obvious type of restriction to the form of a KB is what might be called *database form*. Following the structure of what is typically found in Database Management, we imagine a KB divided into two parts. The first contains only a set of function-free ground atoms such as

Cousin(fred,george)	Male(joe)
Cousin(fred,wendy)	Male(jim)
Cousin(henry,joe)	Male(fred)
⋮	⋮

This tabular format allows positive instances of various predicates to be characterized. The second part of the KB is a collection of sentences called "integrity constraints" that are used to determine if the first part is reasonable (but not to infer new relationships). For example, the sentence

$$\forall x \forall y \text{Mother}(x, y) \equiv \text{Female}(y) \wedge \text{Parent}(x, y)$$

could be used to rule out a KB where a mother was not also a parent.

To the extent that all we were interested in was the stored "data", this would be the complete database. In this case, however, we would not have queries like

How many cousins does Fred have?

but one more typical of Database Management like

How many tuples in the Cousin *relation contain* fred *in the first column?*

[9] The use of FOL to capture *terminology* or laws is somewhat different. See [7] for details.

The answers here may be different since while there are exactly two *Cousin* tuples, there is nothing so far in the KB that implies that Fred has at least two cousins (since "george" and "wendy" could be names of the same individual) nor that Fred has at most two cousins (since there could be cousins other than those mentioned in the KB). So if we want to use this KB to answer questions in a way that is at least compatible with the database approach, we have to add additional facts to the KB. First of all, it should contain (at least implicitly) all sentences of the form

$$c_i \neq c_j, \qquad \text{for distinct constants } c_i \text{ and } c_j,$$

stating that each constant represents a unique individual. In addition, it also needs for each predicate in the KB a sentence of the form

$$\forall x[\text{Male}(x) \supset x = \text{joe} \vee \cdots \vee x = \text{fred}]$$

telling it that the only instances of the predicate are the ones it has explicitly.[10] With this implicit KB added to the above explicit one, a KR system could, in fact, conclude that Fred has exactly two cousins, just like its Database Management counterpart.

The main property of a KB in this form (including, that is, both its explicit and implicit parts) is that it is much easier than in the general case to answer questions about the world. In particular, since the explicit KB does not use negation, disjunction or existential quantification (except in the integrity constraints), we know the exact instances of every predicate of interest in the language. There is no incompleteness in our knowledge at all. Because of this, *inference reduces to calculation*. To find out how many cousins Fred has, all we have to do is count how many appropriate tuples appear in the *Cousin* relation. We do not, for instance, have to reason by cases or by contradiction, as we would have to otherwise. For example, if we also knew that either Mary or Joe or both was a cousin of Fred but that no cousin of Fred apart from Wendy was female, we could still determine that Fred had three cousins, but not by simply counting. But a KB that is in database form does not allow us to express this kind of uncertainty and, because of this expressive limitation, the KR service is much more tractable.

This limitation on the logical form of a KB has other interesting features. Essentially, what it amounts to is making sure that there is very close structural correspondence between the (explicit) KB and the domain of interest: for each entity in the domain, there is a unique representational object that stands for it; for each relationship that it participates in, there is a a tuple in the KB that corresponds to it. In a very real sense, the KB is an *analogue* of the domain of interest, not so different from other analogues such as maps or physical models. The main advantange of having such an analogue is that it can be used directly to answer questions about the domain. That is, the calculations on the model itself can play the role of more general reasoning techniques much the way arithmetic can replace reasoning with Peano's axioms. The disadvantage of an analogue, however, should also be clear: within a certain descriptive language, it does not allow anything to be left unsaid about the domain.[11] In this sense, an analogue representation can be viewed as a special case of a propositional one where the information it contains is relatively complete.

3.3. Logic Program Form

The second restriction on the form of a KB we will consider is a generalization of the previous one that is found in programs written in PROLOG, PLANNER, and related languages. A KB in logic program form also has an explicit and an implicit part. The explicit KB in a PROLOG program is a collection of first-order sentences (called Horn sentences) of the form

$$\forall x_1 \cdots x_n[P_1 \wedge \cdots \wedge P_m \supset P_{m+1}] \qquad \text{where}$$
$$m \geq 0 \text{ and each } P_i \text{ is atomic.}$$

In the case where $m = 0$ and the arguments to the predicates are all constants, the logic program form coincides with the database form. Otherwise, because of the possible nesting of functions, the set of relevant terms (whose technical name is the Herbrand universe) is much larger and may be infinite.

As in the database case, if we were only interested in the universe of terms, the explicit KB would be sufficient. To understand the KB as being about the

[10] This is one form of what has been called the *closed world assumption* [8].

[11] The same is true for the standard analogues. One of the things a map does not allow you to say, for example, is that a river passes through one of two widely separated towns, without specifying which. Similarly, a plastic model of a ship cannot tell us that the ship it represents does not have two smokestacks, without also telling us how many it does have. This is not to say that there is no *uncertainty* associated with an analogue, but that this uncertainty is due to the coarseness of the analogue (e.g., how carefully the map is drawn) rather than to its content.

world, but in a way that is compatible with the answers provided by a PROLOG processor, we again have to include additional facts in an implicit KB. In this case, the implicit KB is normally infinite since it must contain a set of sentences of the form $(s \neq t)$, for any two distinct terms in the Herbrand universe. As in the database case, it must also contain a version of the closed world assumption which is now a set containing the negation of every ground atomic sentence not implied by the Horn sentences in the explicit KB.

The net result of these restrictions is a KB that once again has complete knowledge of the world (within a given language), but this time, may require inference to answer questions.[12] The reasoning in this case, is the *execution* of the logic program. For example, given an explicit PROLOG KB consisting of

```
parent(bill,mary).
parent(bill,sam).
mother(X,Y) :-
    parent(X,Y), female(Y).
female(mary).
```

we know exactly who the mother of Bill is, but only after having executed the program.

In one sense, the logic program form does not provide any computational advantage to a reasoning system since determining what is in the implicit KB is, in general, undecidable.[13] On the other hand, the form is much more manageable than in the general case since the necessary inference can be split very nicely into two components: a *retrieval* component that extracts (atomic) facts from a database by pattern-matching and a *search* component that tries to use the non-atomic Horn sentences to complete the inference. In actual systems like PROLOG and PLANNER, moreover, the search component is partially under user control, giving him the ability to incorporate some of the kinds of procedural knowledge (or combinatoric advice) referred to earlier. The only purely automatic inference is the retrieval component.

This suggests a different way of looking at the inferential service provided by a KR system (without even taking into account the logical form of the KB). Instead of automatically performing the full deduction necessary to answer questions, a KR system could manage a *limited form of inference* and leave to the rest of the knowledge-based system (or to the user) the responsibility of intelligently completing the inference. As suggested in [9], the idea is to take the "muscle" out of the automatic component and leave the difficult part of reasoning as a problem that the overall system can (meta-)reason about and plan to solve [10].

While this is certainly a promising approach, especially for a KB of a fully general logical form, it does have its problems. First of all, it is far from clear what primitives should be available to a program to extend the reasoning performed by the KR subsystem. It is not as if it were a simple matter to generalize the meager PROLOG control facilities to handle a general theorem prover, for example.[14] The search space in this case seems to be much more complex.

Moreover, it is not clear what the KR service itself should be. If all a KR utility does is perform explicit retrieval over sentences in a KB, it would not be much help. For example, if asked about $(p \vee q)$, it would fail if it only had $(q \vee p)$ in the KB. What we really need is an automatic inferential service that lies somewhere between simple retrieval and full logical inference. But finding such a service that can be motivated semantically (the way logical deduction is) and defined independently of the how any program actually operates is a non-trivial matter, though one we have taken some steps towards in [12].

3.4. Semantic Network Form

Turning now to semantic networks, a first observation about a KB in this form is that it only contains unary and binary predicates. For example, instead of representing the fact that John's grade in cs100 was 85 by

Grade(john, cs100, 85),

we would postulate the existence of objects called "grade-assignments" and represent the fact about John in terms of a particular grade-assignment e as

$$\text{Grade-assignment}(e) \wedge \text{Student}(e, \text{john}) \wedge \\ \text{Course}(e, \text{cs100}) \wedge \text{Mark}(e, 85).$$

This part of a KB in semantic net form is also in database form: a collection of function-free ground atoms, sentences stating the uniqueness of constants and the closed world assumption.

[12] Notice that it is impossible to state in a KB of this form that $(p \vee q)$ is true without stating which, or that $\exists x P(x)$ is true without saying what that x is. However, see the comments below regarding the use of encodings.

[13] In other words, determining if a ground atomic sentence is implied by a collection of Horn sentences is undecidable.

[14] Though see [11] for some ideas in this direction.

The main feature of a semantic net (and of the frame form below), however, is not how individuals are handled, but the treatment of the unary predicates (which we will call *types*) and binary ones (which we will call *attributes*). First of all, the types are organized into a taxonomy, which, for our purposes, can be represented by a set of sentences of the form[15]

$$\forall x[B(x) \supset A(x)].$$

The second kind of sentence in the generic KB places a constraint on an attribute as it applies to instances of a type:

$$\forall x[B(x) \supset \exists y(R(x,y) \land V(y))] \quad \text{or}$$
$$\forall x[B(x) \supset R(x,c)].^{16}$$

This completes the semantic net form.

One property of a KB in this form is that it can be represented by a labelled directed graph (and displayed in the usual way). The nodes are either constants or types, and the edges are either labelled with an attribute or with the special label *is*-a.[17] The significance of this graphical representation is that it allows certain kinds of inference to be performed by simple graph-searching techniques. For example, to find out if a particular individual has a certain attribute, it is sufficient to search from the constant representing that individual, up *is*-a links, for a node having an edge labelled with the attribute. By placing the attribute as high as possible in the taxonomy, all individuals below it can *inherit* the property. Computationally, any mechanism that speeds up this type of graph-searching can be used directly to improve the performance of inference in a KB of this form.

In addition, the graph representation suggests different kinds of inference that are based more directly on the structure of the KB than on its logical content. For example, we can ask how two nodes are related and answer by finding a path in the graph between them. Given for instance, Clyde the elephant and Jimmy Carter, we could end up with an answer saying that Clyde is an elephant and that the favourite food of elephants is peanuts which is also the major product of the farm owned by Jimmy Carter. A typical method of producing this answer would be to perform a "spreading activation" search beginning at the nodes for Clyde and Jimmy. Obviously, this form of question would be very difficult to answer for a KB that was not in semantic net form.

For better or worse, the appeal of the graphical nature of semantic nets has lead to forms of reasoning (such as default reasoning [15]) that do not fall into standard logical categories and are not yet very well understood [16].[18] This is a case of a representational notation taking on a life of its own and motivating a completely different style of use not necessarily grounded in a truth theory. It is unfortunately much easier to develop algorithms that appear to reason over structures of a certain kind than to *justify* its reasoning by explaining what the structures are saying about the world.

This is not to say that defaults are not a crucial part of our knowledge about the world. Indeed, the ability to abandon a troublesome or unsuccessful line of reasoning in favour of a default answer seems intuitively to be a fundamental way of coping with incomplete knowledge in the presence of resource limitations. The problem is to make this intuition precise. Paradoxically, the best formal accounts we have of defaults (such as [17]) would claim that reasoning with them is even *more difficult* than reasoning without them, so research remains to be done.

One final observation concerns the elimination of higher arity predicates in semantic networks. It seems to be a fairly common phenomenon that a certain generality of logical form can be avoided by introducing special representational objects into the domain. In the example above, a special "grade-assignment" object took the place of a 3-place predicate. Another example is the use of encodings of sentences as a way of providing (what appears to be) a completely extensional version of modal logic [18].[19] Not that exactly the same expressiveness is preserved in these cases; but what *is* preserved is still fairly mysterious and

[15] See [13] for a discussion of some of the subtleties involved here.

[16] There are other forms possible for this constraint. For example, we might want to say that *every R* rather than *some R* is a V. See also [14]. For the variant we have here, however, note that the KB is no longer in logic program form.

[17] Note that the interpretation of an edge depends on whether its source and target are constants or types. For example, from a constant c to a type B, *is*-a says $B(c)$, but from a type B to a type A, it is a taxonomic sentence (again, see [13]).

[18] A simple example of a default would be to make *elephant* have the colour *grey* but to allow anything below *elephant* (such as *albino-elephant*) to be linked to a different colour value. To determine the colour of an individual would involve searching up for a value and stopping when the first one is found, allowing it to preempt any higher ones.

[19] Indeed, some modern semantic network formalisms (such as [19]) actually include all of FOL by encoding sentences as terms.

deserves serious investigation, especially given its potential impact on the tractability of inference.

3.5. Frame Description Form

The final form we will consider, the frame description form, is mainly an elaboration of the semantic network one. The emphasis, in this case, is on the structure of types themselves (usually called *frames*) in terms of their attributes (called *slots*). Typically, the kind of detail that is involved includes

1. *values*, stating exactly what the attribute of an instance should be. Alternatively, the value may be just a *default*, in which case an individual inherits the value provided he does not override it.
2. *restrictions*, stating what constraints must be satisfied by attribute values. These can be *value* restrictions, specified by a type that attribute values should be instances of, or *number* restrictions, specified in terms of a minimum and a maximum number of attribute values.
3. *attached procedures*, providing procedural advice on how the attribute should be used. An *if-needed* procedure says how to calculate attribute values if none have been specified; an *if-added* procedure says what should be done when a new value is discovered.

Like semantic networks, frame languages tend to take liberties with logical form and the developers of these languages have been notoriously lax in characterizing their truth theories [14]. What *we* can do, however, is restrict ourselves to a non-controversial subset of a frame language that supports descriptions of the following form:

```
(Student
    with a dept is computer-science and
    with ≥ 3 enrolled-course is a
      (Graduate-Course
          with a dept is a
            Engineering-Department))
```

This is intended to be a structured type that describes Computer Science students taking at least three graduate courses in departments within Engineering. If this type had a name (say A), we could express the type in FOL by a "meaning postulate" of the form

$$\begin{aligned}&\forall x A(x) \equiv\\ &\quad [\text{Student}(x) \wedge \text{dept}(x, \text{computer-science}) \wedge\\ &\qquad \exists y_1 y_2 y_3\ (y_1 \neq y_2 \wedge y_1 \neq y_3 \wedge y_2 \neq y_3 \wedge\\ &\qquad\qquad \ldots y_1 \ldots \wedge\\ &\qquad\qquad \ldots y_2 \ldots \wedge\\ &\qquad\qquad \ldots y_3 \ldots)].\end{aligned}$$

Similarly, it should be clear how to state equally clumsily[20] in FOL that an individual is an instance of this type.

One interesting property of these structured types is that we do not have to state explicitly when one of them is below another in the taxonomy. The descriptions themselves implicitly define a taxonomy of *subsumption*, where type A subsumes type B if, by virtue of the form of A and B, every instance of B must be an instance of A. For example, without any world knowledge, we can determine that the type *Person* subsumes

(Person with every male friend is a Doctor)

which in turn subsumes

(Person with every friend is a
(Doctor with a specialty is surgery)).

Similarly,

(Person with ≥ 2 children) subsumes
(Person with ≥ 3 male children).

Also, we might say that two types are *disjoint* if no instance of one can be an instance of the other. An example of disjoint types is

(Person with ≥ 3 young children) and
(Person with ≤ 2 children).

Analytic relationships like subsumption and disjointness are a property of structured types that is not availble in a semantic net where all of the types are atomic.

There are very good reasons to be interested in these analytic relationships [7]. In KRYPTON [4], a full first-order KB is used to represent facts about the world, but subsumption and disjointness information is also available without having to add to the KB a collection of meaning postulates representing the structure of the types. The reason this is significant is that while subsumption and disjointness can be defined in

[20] What makes these sentences especially awkward in FOL is the number restrictions. For example, the sentence "*There are a hundred billion stars in the Milky Way Galaxy*" would be translated into an FOL sentence with on the order of 10^{22} conjuncts.

terms of logical implication,[21] there are good special-purpose algorithms for calculating these relationships in frame description languages. Again, because the logical form is sufficiently constrained, the required inference is much more tractable.

As it turns out, frame description languages are themselves a microcosm of the tradeoff between expressiveness and tractability. If we imagine a language of structured descriptions specified by a grammar of types and attributes, we get a family of dialects by including or omitting certain formation rules.[22] In one dialect, we might allow a description to place a maximum on the number of attribute values, for instance, but not a minimum. As discussed in [20], the remarkable property of these dialects is that a small increase in expressive power can make subsumption completely intractable. To keep subsumption manageable, the inclusion of certain formation rules has to be compensated by the omission of others. Because it is relatively easy to cross the threshold from the tractable to the intractable, structured description languages seem to be a good place to study the tradeoff between expressivenes and tractability, and perhaps even to discover the fundamental parameters that control it.

4. Conclusions and Morals

In this final section, we step back from the details of the specific representational formalisms we have examined and attempt to draw a few conclusions.

An important observation about these formalisms is that we cannot really say that one is *better* than any other; they simply take different positions on the tradeoff between expressiveness and tractability. For example, full FOL is both more expressive and less appealing computationally than a language in semantic net form. Nor is it reasonable to say that expressiveness is the primary issue and that the other is "merely" one of efficiency. In fact, we are not really talking about efficiency here at all; that, presumably, is an issue of algorithm and data structure, concerns of the Symbol Level [1]. The tractability concern we have here is much deeper and involves whether or not it makes sense to even think of the language as computationally based.

From the point of view of those doing research in KR, this has a very important consequence: we should continue to design and examine representation languages, *even when these languages can be viewed as special cases of FOL.* What really counts is that these special cases be interesting both from the point of view of what they can represent, and from the point of view of the reasoning strategies they permit. All of the formalisms we have examined above satisfy these two requirements. To dismiss a language as *just* a subset of FOL is probably as misleading as dismissing the notion of a context-free grammar as just a special case of a context-sensitive one.

What truth in advertising does require, however, is that these special cases of FOL be identified as such. Apart from allowing a systematic comparison of representation languages (as positions on the tradeoff), this might also encourage us to consider systems that use more than one sublanguage and reasoning mechanism (as suggested for equality in [21]). The KRYPTON language, for example, includes all of FOL *and* a frame description language. To do the necessary reasoning, the system contains both a theorem prover and a description subsumption mechanism, even though the former could do the job of the latter (but much less efficiently). The trick with these *hybrid systems* is to factor the reasoning task so that the specialists are able to cooperate and apply their optimized algorithms without interfering with each other.

These considerations for designers of representation languages apply in a similar way to those interested in populating a KB with a theory of some sort. A good first step might be to write down a set of first-order sentences characterizing the domain, but it is somewhat naive to stop there and claim that the account could be made computational after the fact by the inclusion of a theorem prover and a few well chosen heuristics. What is really needed is the (much more difficult) analysis of the logical form of the theory, keeping the tradeoff clearly in mind. An excellent example of this is the representation of *time* described in [22]. Allen is very careful to point out what kind of information about time cannot be represented in his system, as well as the computational advantage he gains from this limitation.

For the future, we still have a lot to learn about the tradeoff. It would be very helpful to accumu-

[21] Specifically, type A subsumes type B iff the meaning postulates for A and B logically imply $\forall x[B(x) \supset A(x)]$.

[22] The fact that these languages have a simple *structural* form (as exhibited by formation rules) does not mean that they have a simple *logical* form (as exhibited by meaning postulates), however.

late a wide variety of data points involving tractable and intractable languages. Especially significant are crossover points where small changes in a language change its computational character completely. Moreover, we need to know more about what *people* find easy or hard to handle. There is no doubt that people can reason when necessary with radically incomplete knowledge (such as that expressible in full FOL) but apparently only by going into a special problem-solving or logic puzzle mode. In normal common sense situations, when reading a geography book, for instance, the ability to handle disjunctions (say) seems to be quite limited. The question is what forms of incomplete knowledge *can* be handled readily, given that the geography book is not likely to contain any procedural advice on how to reason.

In summary, we feel that there are many interesting issues to pursue involving the tradeoff between expressiveness and tractability. Although there has always been a temptation in KR to either set the sights too low (and provide only a data structuring facility with little or no inference) or too high (and provide a full theorem proving facility), this paper argues for the rich world of representation that lies between these two extremes.

ACKNOWLEDGEMENTS

Many of the ideas presented here originally arose in the context of the KRYPTON project. I am deeply indebted to the Fairchild AI Lab for making this research possible and to the other members of the project: Ron Brachman, Richard Fikes, Peter Patel-Schneider, and Victoria Pigman. I would especially like to thank Ron, Peter, and Jim des Rivières for providing very helpful comments on a earlier draft of this paper, and S. J. Hurtubise for not.

REFERENCES

[1] Newell, A., The Knowledge Level, *The AI Magazine*, Vol. 2, No. 2, 1981.

[2] Smith, B. C., *Reflection and Semantics in a Procedural Language*, Ph.D. thesis and Tech. Report MIT/LCS/TR-272, MIT, Cambridge, MA, 1982.

[3] Levesque, H. J., Foundations of a Functional Approach to Knowledge Representation, *Artificial Intelligence*, forthcoming.

[4] Brachman, R. J., Fikes, R. E., and Levesque, H. J., KRYPTON: A Functional Approach to Knowledge Representation, *IEEE Computer*, October, 1983.

[5] Moore, R. C., The Role of Logic in Knowledge Representation and Commonsense Reasoning, *Proc. AAAI-82* Pittsburgh, PA 1982, 428–433.

[6] Levesque, H. J., *A Formal Treatment of Incomplete Knowledge Bases*, Technical Report No. 3, Fairchild Laboratory for Artificial Intelligence Research, Palo Alto, CA, 1982.

[7] Brachman, R. J., and Levesque, H. J., Competence in Knowledge Representation, *Proc. AAAI-82*, Pittsburgh, PA, 1982, 189–192.

[8] Reiter, R., On Closed World Data Bases, In *Logic and Data Bases*, Gallaire, H. and Minker, J. (eds.), Plenum Press, New York, 1978.

[9] Frisch, A. and Allen, J., Knowledge Representation and Retrieval for Natural Language Processing, TR 104, Computer Science Dept., U. of Rochester, 1982.

[10] Genesereth, M. R., An Overview of Meta-Level Architecture, *Proc. AAAI 83*, Washington, D. C., 1983.

[11] Stickel, M. E., A Prolog Technology Theorem Prover, *Proc. of the 1984 Symposium on Logical Programming*, Atlantic City, 1984.

[12] Levesque, H. J., A Logic of Knowledge and Active Belief, submitted to AAAI-84, Austin, Texas, 1984.

[13] Brachman, R. J., What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks, *IEEE Computer*, October, 1983.

[14] Hayes, P. J., The Logic of Frames, in *Frame Conceptions and Text Understanding* Metzing, D., (ed.), Walter de Gruyter and Co., Berlin, 1979, 46–61.

[15] Reiter, R., On Reasoning by Default, *Proc. TINLAP-2*, University of Illinois at Urbana-Champaign, 1978.

[16] Etherington, D., and Reiter, R., On Inheritance Hierarchies with Exceptions, *Proc. AAAI-83*, Washington, D. C., 1983.

[17] Reiter, R., A Logic for Default Reasoning, *Artificial Intelligence*, Vol. 13, No. 1,2, 1980, 81–132.

[18] Moore, R., *Reasoning about Knowledge and Action*, Technical Note 191, SRI International, Menlo Park, 1980.

[19] Shapiro, S. C., The SNePS Semantic Network Processing System, in *Associative Networks: Representation and Use of Knowledge by Computers*, N. V. Findler (ed.), Academic Press, New York, 1979, 179–203.

[20] Brachman, R. J. and Levesque, H. J., The Tractability of Subsumption in Frame-Based Description Languages, submitted to AAAI-84, Austin, Texas, 1984.

[21] Nelson, G. and Oppen, D. C., Simplification by Cooperating Decision Procedures, *ACM Transactions on Programming Languages and Systems*, Vol. 1, No. 2, 1979, 245–257.

[22] Allen, J., Maintaining Knowledge about Temporal Intervals, *Communications of the ACM*, Vol. 26, No. 11, 1983.

Representing Control Strategies Using Reflection

Bryan M. Kramer
Department of Computer Science
University of Toronto
Toronto, Ontario
M5S 1A4

Abstract. Control or strategic knowledge is necessary to enable knowledge based systems containing large numbers of facts to choose appropriately among these facts. In this paper, an architecture is presented that allows such knowledge to be stated in the same language that is used to express facts about the domain. This architecture is obtained by creating a reflective ([12]) architecture based on a Horn clause processor.

Introduction

Much of the power of a knowledge-based expert system lies in its ability to apply a large amount of knowledge to a variety of problems. However, as the amount of knowledge grows, an increasingly sophisticated inference engine or control component is necessary to extract only the facts relevant to a problem at hand.

Past research has suggested that one way of providing sophisticated control is through an architecture which includes a knowledge base of facts pertaining to control ([2,13]). Indeed, it has been argued that, given that an expert system must acquire its knowledge through interaction with experts, and that experts' knowledge consists to a large degree of control strategies, that it is essential that this knowledge be represented in a knowledge base ([13]).

In this paper an architecture based on a combination of a Horn clause knowledge base and Smith's concept of reflection ([12]) is presented. It is then shown how various schemes for controlling deduction currently used in knowledge based systems can be expressed in the framework of this architecture.

Control in Knowledge-Based Systems

A major motivation for the use of a knowledge base of strategic knowledge is, of course, the potential improvement in the efficiency of the system. There are however, other strong motivations.

Firstly, knowledge bases are designed to be easy to modify. When control strategies are stored in such a knowledge base, the techniques used to acquire factual knowledge can also be applied to the acquisition of strategic knowledge. In systems where control is held by an inference engine coded in some programming language, the expert would have to communicate his strategies to a programmer who would have to modify the program. For example, in a system to give investment advice described by Davis ([2]), an expert might want the system to know that "when dealing with older subjects avoid facts which would support conclusions advising high risk investments". This rule captures the knowledge that deductions for older subjects will never result in suggestions for high risk investments, and that the system should therefore avoid bad search paths by avoiding certain classes of knowledge. If such rules are stored in a knowledge base, it is just as easy to acquire them from the expert as it is to acquire the fact that "Company x is a high risk investment".

Secondly, programs can use the control knowledge base as data and reason about it or with it. This is important for the explanation component of the expert system --- in the traditional architecture the explanation component would have to be modified each time the inference engine were changed. More importantly, one can use an inference engine which can deduce from the strategic knowledge and the state of the problem solution the best strategy to follow at a given time. In addition, a program can use the strategic knowledge base and the history of various processes to deduce new strategies. This is illustrated by Lenat's AM system ([9]).

The above are examples of the use of knowledge in more than one way. This is, however, a major motivation for the use of a knowledge representation language to express the factual knowledge of a domain ([11]). Similarly, the representation of control would equally benefit from the abstraction and organizational mechanisms provided by these languages. It therefore makes sense to apply Hayes' mandate ([7]) and use the same language to represent both factual knowledge and strategic knowledge.

A Reflective Horn Clause Architecture

Expressing Facts

Horn clauses as in ([8]) are used as the knowledge base language. Horn clauses represent knowledge as implications having one consequent which follows from the conjunction of zero or more antecedents. All variables are universally quantified. An example, in the notation used in this paper, is

```
(Uncle $uncle $person) <=
      (Father $father $person)
      (Brother $father $uncle)
```

Atoms such as "$uncle" are variables. The clause states that for any three people "$father", "$uncle", and "person", if "$father" is the father of "$person" and the brother of "$uncle" then "$uncle" is the uncle of "$person".

The knowledge used to make control decisions consists of facts about the clauses in the knowledge base. These facts must be represented as clauses which contain constant symbols which refer to other clauses in the knowledge base. Control knowledge can also refer to the individual terms (consequents and antecedents) of other clauses. The following is an example of facts that might be known about a clause:

```
(clause:consequent %c1 (P '$x '$y)) <=

(clause:antecedents %c1 %a1.1) <=

(ante:first %a1.1 (Q '$x '$y '$z)) <=

(ante:rest %a1.1 %a1.2) <=

(ante:first %a1.2 (R '$z)) <=

(ante:rest %a1.2 nil) <=

(priority %c1 9) <=
```

The special constant symbols that start with "%" are used to name objects that name clauses and parts of clauses. The notation "'$z" names the variable "$z". In the example, "%c1" is a constant symbol representing a clause with consequent "(P $x $y)" and antecedents "(Q $x $y $z) (R $z)" having priority 9. The predicates ante:first and ante:rest represent a list of antecedents. Note that there is no special meaning in the predicate names containing a colon; this is merely a convention used to group related predicates. Thus "clause:consequent" and "clause:antecedents" are two predicates that have in common the fact that the first argument is a clause. Note also that the first six clauses provide a representation of the clause; that is, the same information that is contained in the representation

```
(P $x $y) <= (Q $x $y $z) (R $z)
```

while the clause containing the predicate "priority" is an additional fact about the clause. This latter fact might be useful in controlling a deduction which uses "%c1".

The inference engine on which the architecture is based is a simple backward chaining interpreter of the clauses. This inference engine maintains a set of goals which are terms of Horn clauses with values (which may contain other variables) for the variables. Inference proceeds by choosing one of these goals, finding a clause whose consequent unifies with that goal, and replacing that goal with the antecedents of the matching clause with an assignment of values to the variables as specified by the unification. If there is no matching clause it is necessary to backtrack to an earlier state in the sequence of inferences and choose an alternative matching clause.

Reflection

Brian Smith's procedural reflection arises from what he calls the process reduction model of computation. In this model, a process can be decomposed into another process called a processor acting on a procedure represented in the machine. Thus an accounting process implemented in LISP can be decomposed into a LISP processor and a LISP procedure represented in the machine by cons cells. Similarly, the same accounting process might have been implemented as a set of Horn clauses and an processor for Horn clauses. In this case the set of Horn clauses is a knowledge base and the processor is the inference engine.

Smith then notes that the decomposition of a process can be continued by decomposing the processor. For example, the LISP processor could be viewed as a processor for some high level language executing a program text which describes a LISP processor. The interesting case is that in which the processor for a language A is decomposed into a processor for the language A and a text in language A describing that processor. This is achieved, for example, when one applies LISP's eval to a representation of itself and some LISP program. Smith calls the representation of the processor in the language it is processing a meta-circular processor. Similarly, one can write a set of Horn clauses which when processed by a Horn clause processor itself acts as a Horn clause processor. Kowalski ([8]) gives an example of such an processor.

Consider a situation in which a Horn clause processor, P, is processing a a Horn clause knowledge base. In particular, let M be a subset of this knowledge base which implements a meta-circular processor for Horn clauses, and assume that the goals maintained by P are terms of the clauses of M. Thus the processor P is in effect executing the meta-circular processor, M, which will be therefore be processing a subset H of the knowledge base. Now suppose a clause C which is not in M matches a goal in P's processing of M. This clause will represent some fact about the clauses H and the state of the processor (implemented by M) processing them. If, furthermore, control returns to M after the processing of C, some processing will have occurred which might affect the continued execution of M. A system supports procedural reflection when it allows the augmentation of its meta-circular processor M with clauses such as C. Notice that the ability to augment the knowledge base with clauses which reason with the program and the state of the process is exactly what is required a system in which control knowledge is represented.

This model can be extended to allow arbitrary decomposition of the processor. In effect, the system can be viewed as an infinite tower of meta-circular processors each processing the one below, that is, H is processed by M which is processed by M, etc. At any level clauses from the knowledge base might affect the course of execution.

The architecture for controlled knowledge bases is based on such a model. As in Smith, reflection is restricted to significant points in the execution cycle of the processor. In Smith's 3-LISP reflection occurs at the point when the processor looks at the function position of an s-expression: if the function is one that is designated reflective, its body is run at the level of the processor. An analagous Horn clause system would be one in which certain predicates are designated reflective and when a goal using such a predicate is encountered, its antecedents become goals of the processor of the processor. In the Horn clause architecture of this paper, however, reflection can occur at more points in the control cycle. Moreover, these points are those at which knowledge supplied by the expert is useful to make control decisions. At significant places in the control cycle, the processor of the processor attempts to match goals of the form

```
   (control <arguments> <name of place in control
      cycle>) <=
         <body>
```

which if matched indicate a reflective situation where the body is processed at the level of the processor. For example, suppose "%a1.1" is the goal which has been matched against the clause knowledge base; then if the clause

```
   (control $history $conflict-set $node %a1.1 $clause
      CHOOSE-MATCH) <=
         <body>
```

is in the clause knowledge base at the level of the processor, <body> will be executed at the level of the processor.

Smith's restriction on the origin of reflective activity is what makes possible an implementation of an architecture containing an arbitrary number of levels. Since reflective activity arises from objects encountered in the processing from the bottom up, at most a finite number of levels can be affected by reflective code at any time. The implementation can therefore consist of a processor P that simulates an infinite tower of meta-circular processors in which no reflection occurs. It would be P that processes a copy of the meta-circular processor plus reflective code, M+, which processes another M+, and so on until a copy processes H, the Horn clause program. If no reflection has occurred, P can process H directly. Also, P can be written so that, should it detect that the M+ below it no longer is affected by reflective code it can absorb that processor and run one level closer to H. When a control clause such as the one above is encountered, a new copy of M plus the relevant control clauses is added between P and the previous M+.

A Meta-Circular Processor

The state of a process is described by a tree of nodes containing the history of the process. Each node contains a set of goals that have not yet been executed. A child is derived from its parent by the inference rule: a goal is chosen, a clause whose consequent unifies with that goal is found, and the child's set of goals consists of the goals of the parent in which the goal that was unified is replaced by the antecedents of the unifying clause. A node is marked open if it contains goals that may still unify with clauses in the knowledge base. It is marked success if it contains the empty set of goals, and failure if all its children are marked failure and no further children can be generated. This processor differs from one such as Kowalski's ([8]) in that the representation of the history allows explicit control of backtracking.

Given this structure, the following choice points can be identified in the inference algorithm: 1) choosing a node in the history tree. Two uses for this are: a) locating a place in the tree to resume processing should a leaf fail (backtracking), and b) switching from node to node in successive execution cycles to pursue more than one line of reasoning at a time. 2) choosing a goal in the node. Choosing a most recently included goal is depth first processing while choosing an oldest goal is breadth first processing. 3) choice of a clause context. A common way of making matching more efficient is to attempt to match only a subset of the clause database. 4) choice of a member of the conflict set. The algorithm generates one new node at a time. However, the search for matching clauses will return all those that match. Conflict resolution involves choosing among these for a single clause to use. OPS5 ([10]) has a complicated conflict resolution algorithm involving how recently a clause was added to the knowledge base, statistics about the goal that is matched, etc. Note that a simpler system could avoid this step by using step 3) to choose orderings of clauses and then use "take the first match" as a conflict resolution strategy. However, it might be the case that decisions can be made based on properties of the conflict set as a whole.

The following is a sketch of a meta-circular processor which allows reflective reasoning at these choice points. The predicate "solve" corresponds to Kowalski's "Demo". The details of procedures such as "find-unifying-clause" are suppressed.

```
  (solve $history $clauses $result) <=
        (choose-node $history $clauses $node)
        (choose-goal $node $clauses $goal)
        (choose-clause $node $goal $clauses $new-
           clauses)
        (find-unifying-clause $node $goal $new-
           clauses $conflict-set)
        (choose-match $history $conflict-set $node
           $goal $clause)
        (create-leaf $history $node $goal $clause
           $new-history)
        (solve $new-history $new-clauses $result)

  (choose-node $history $clauses $node) <=
        (control $history $clauses $node CHOOSE-NODE)

  (choose-node $history $clauses $node) <=
        (default-choose-node $history $clauses $node)

  (choose-goal $node $clauses $goal) <=
        (control $node $clauses $goal CHOOSE-GOAL)

  (choose-goal $node $clauses $goal) <=
        (default-choose-goal $node $clauses $goal)
```

```
(choose-match $history NIL $node $goal $clause) <=
      (mark-node $node FAIL)

(choose-match $history $conflict-set $node $goal
   $clause) <=
      (control $history $conflict-set $node $goal
         $clause CHOOSE-MATCH)

(choose-match $history $conflict-set $node $goal
   $clause) <=
      (default-choose-match $history $conflict-set
         $node $clause)

(choose-clause $node $goal $clauses $new-clauses)
   <=
      (control $node $goal $clauses $new-clauses
         CHOOSE-CLAUSES)

(choose-clause $node $goal $clauses $new-clauses)
   <=
      (default-choose-clause $node $goal $clauses
         $new-clauses)
```

The explicit choice points are rules with the heads "(choose-<x> ...)" where x names the choice points. If no control clause is located, the backtracking results in the execution of the default case. In general a control goal is invoked with bindings for all but the last variable, and the execution of the body is expected to bind the last variable with a value which will be used in the next step of the control cycle. For example, a control goal of type "CHOOSE-MATCH" provides a goal tree, a conflict set, a node, and a goal, and is expected to bind to "$clause" a clause from those in the conflict set which will be used by "create-leaf" to create a new node.

The processor which executes this meta-circular processor takes as its default control decisions the following: goals are expanded depth first from left to right (in the textual representation), conflicts are resolved by taking the first match encountered in reading the text, and backtracking occurs at the most recent choice.

An important feature of the architecture is that clauses containing the term "(assert <a-clause>)" result in the addition of the clause "<a-clause>" to the knowledge base. This enables the meta-circular processor and control clauses in the knowledge base to record the state of a series of inferences for use in subsequent control cycles. For example, reflective code might want to assert additional facts about nodes in the control tree which subsequent calls to control clauses of the type "CHOOSE-NODE" might use.

Applications

The architecture above allows strategic knowledge to be expressed as Horn clauses. Note that, although control results from clauses containing the control predicate, the antecedents can involve arbitrary predicates and arbitrarily complicated reasoning involving knowledge about clauses, goals, nodes, etc. Examples of such meta-knowledge include simple facts such as

```
(priority %c109 10) <=
```

which assigns a priority to a clause which could be used to by the body of a control clause for conflict resolution, or more complicated implications such as

```
(relevancy %c122 $g low) <=
      (type $g high-risk)
```

which states that a certain clause is not very relevant to high risk goals. This section illustrates how control clauses and meta-knowledge can be used to express various schemes for controlling deduction.

First, the ability to choose a node in the history tree allows sophisticated backtracking schemes. For example, one could write control rules which record the dependencies between goals and rules, that is, rules that record the fact that a given goal appears in a given node due to the application of some particular clause to some other node. The body of a "CHOOSE-NODE" control rule could then return the node where that inference occurred, and thereby implement dependency directed backtracking as, for example, in AMORD ([14]).

Secondly, choice of a goal is important for the efficiency of an expert system. In the reflective architecture, the processing invoked through the bodies of control rules allow arbitrarily complicated processing to choose a goal. Through a rule such as

```
(control $node $rules $goal CHOOSE-GOAL) <=
      (goal-planner ....)
```

one can invoke a planner such as that described by Corkill and Lesser ([1]) for their goal directed HEARSAY-II architecture.

The goal selection choice point also corresponds to the Hearsay-III architecture described in ([3]). The set of active goals in the current node compares with the Hearsay-III control blackboard, and control rules which are invoked by the presence of certain goals in the current node compare with control knowledge sources.

As mentioned above, the choose clause choice point allows the implementation of clause set partitioning. An example of this is Georgeff's scheme ([6]) in which the control knowledge consists of constraints on the sequence of invocation of clauses as expressed, for example, as a regular expression:

```
%c1, (%c2, %c3, %c4) *, %c5
```

The "%ci" name clauses, and the regular expression expresses the fact that a legal execution consists of a call to "%c1" followed by zero or more sequences of calls to "%c2" then "%c3", then "%c4", followed by a call to "%c5". One of the control clauses which could implement this might be

```
(control $node $goal $clauses {%c2, %c5} CHOOSE-
   CLAUSES) <=
      (last-call %c4)
```

which means that the new clause partition contains "%c2" and "%c5" if the last rule called was "%c4" (assuming a control clause in choose-match records which clause it called).

Finally, choices from the conflict set correspond to the *meta-rules* of Davis' program TIERESIAS ([2]). In this case, the control clause invokes a procedure which chooses from the conflict set the match most likely to succeed. The reflective architecture, in fact, solves a problem mentioned by Davis: in his system, conclusions made at one invocation of the meta-level cannot by used by a subsequent invocation.

Related Work

As mentioned earlier, the concept of procedural reflection as used in this paper comes from the work of Brian Smith. His work involved the creation of a reflective dialect of LISP which he called 3-LISP. This was then used to show how reflection provided a clean mechanism for defining common additions to LISP such as the functions "catch" and "throw".

Secondly, as is also clear in the above, Davis's work with *meta-rules* has strongly influenced the reflective architecture. In fact, the reflective meta-circular processor bears resemblance to his execution cycle. The reflective architecture differs, however, in the addition of locations where meta-level reasoning may occur (i.e. choice of node, etc.), and in the ability of meta-level reasoning to record conclusions to influence meta-level reasoning at subsequent choice points and subsequent execution cycles. In addition, the Horn clause system has the ability to store facts about clauses, nodes, goals etc. (e.g. the relevance of a clause to a goal) that is missing in Davis's system.

Finally, the *meta-level architecture* described in ([4] and ([5] addresses the problem of controlling the problem solving activity of a program. This architecture involves a theorem prover which deduces facts, the operation and arguments, about a desirable action which is the passed to a processor for execution. The meta-level architecture is presented as a general scheme for which decisions such as the nature of the processor and the number of meta-levels must be made for each application. In contrast, the reflective Horn clause architecture has a specific processor which is representable in the knowledge base, and the level changing implementation allows an arbitrary number of reflective levels.

Summary

This paper describes an architecture for expert systems which provides the ability to describe control knowledge in the same language used to describe facts about the world. It has been shown in the past that such control knowledge is necessary in the implementation of efficient and understandable expert systems. The paper illustrates that the reflective architecture nicely captures many control schemes previously proposed.

The architecture has been implemented on a VAX computer using Franz LISP. This implementation consists of a processor which simulates an infinite tower of processors, and a meta-circular processor as described above.

References

[1] Corkill, Daniel D., Lesser, Victor R., and Hudlicka, Eva, "Unifying Data-Directed and Goal-Directed Control," *The Proceedings of the Second National Conference on Artificial Intelligence*, pp.143-147 (August 1982).

[2] Davis, Randall, "Meta-Rules: Reasoning About Control," *Artificial Intelligence* Vol. 15(3), pp.179-222 (Dec. 1980).

[3] Erman, Lee D., London, Philip E., and Fickas, Stephen F., "The Design and an Example Use of HEARSAY-III," *The Proceedings of the 7th IJCAI*, pp.409-415 (1981).

[4] Genesereth, Michael R., "An Overview of Meta-Level Architecture," *The Proceedings of the National Conference on Artificial Intelligence*, pp.119-124 (1983).

[5] Genesereth, Michael R. and Smith, David E., "Meta-Level Architecture," Stanford HPP-81-6, Department of Computer Science, Stanford University (1982).

[6] Georgeff, Michael P., "A Framework for Control in Production Systems," *The Proceedings of the 6th IJCAI*, pp.328-334 (1979).

[7] Hayes, Patrick J., "In Defense of Logic," *The Proceedings of the 5th IJCAI*, pp.559-65 (1977).

[8] Kowalski, Robert, *Logic for Problem Solving*, Elsevier North Holland, New York, New York (1979).

[9] Lenat, Douglas B. and Harris, Gregory, "Designing a Rule Base System that Searches for Scientific Discoveries," pp. 25-51 in *Pattern Directed Inference Systems*, ed. Waterman and Hayes-Roth, Academic Press (1978).

[10] McDermott, J. and Forgy, C., "Production System Conflict Resolution Strategies," pp. 177-199 in *Pattern Directed Inference Systems*, ed. Waterman and Hayes-Roth, Academic Press (1978).

[11] Mylopoulos, John, "An Overview of Knowledge Representation," *SIGART Newsletter*(74), pp.5-12 (January 1981). The Workshop on Data Abstraction Data Bases, and Conceptual Modelling.

[12] Smith, Brian C., "Reflection and Semantics in a Procedural Language," MIT/LCS/TR-272, Massachusetts Institute of Technology, Cambridge, Ma. (1982).

[13] Stefik, Mark, "Planning and Meta-Planning (MOLGEN: Part 2)," Artificial Intelligence Vol. 16(2) (May 1981).

[14] deKleer, John, Doyle, Jon, Steele, Guy L. Jr., and Sussman, Gerald Jay, "AMORD Explicit Control of Reasoning," SIGART(64) (1977).

KNOWLEDGE BASE DESIGN
FOR AN
OPERATING SYSTEM EXPERT CONSULTANT*

Stephen J. Hegner
Department of Computer Science
and Electrical Engineering
Votey Building
University of Vermont
Burlington, VT 05405

Robert J. Douglass
Los Alamos National Laboratory
C-10, MS B296
Los Alamos, NM 87545

Abstract

Yucca is an operating system expert consultant currently under development at Los Alamos National Laboratory and the University of Vermont. It consists of two modules, the *natural-language front end*, which translates natural-language queries to and from the formal query language *Quelya*, and the *knowledge base*, which uses a completely formal representation of the behavior of the operating system to answer the Quelya queries submitted by the front end. In this paper the overall architecture of the knowledge base is outlined.

1. Introduction

Yucca is an operating system expert consultant currently under development at Los Alamos National Laboratory and the University of Vermont. It is designed to provide expert help to experienced computer users who are encountering a new operating system for the first time. To automate such assistance, a utility must go beyond the capabilities of ordinary operating system help facilities in several respects. First of all, the user must not need to know the name of the appropriate command to get help. In other words, the system must be able to answer questions of the "How do I ...?" variety. For example, a user desiring to delete a file must not need to know the name of the delete command in order to get further help. Second, the expert helper must be able to provide information about the static structure of the system; questions of the "What is user may wish to know what a pipe is (in UNIX[a]). Third, the expert helper must have enough knowledge of other systems to be able to provide meaningful responses to questions not directly applicable to the current system. For example, a TOPS-20[b] user may ask a UNIX consultant about online file expiration. Yucca has been designed to take all of these issues into account.

Communication with the user is in natural English. Any other interface would require the user to learn the language of the expert system, which would be self-defeating. Yucca is a utility which exists as an independent subsystem of a currently existing operating system. This is in contradistinction to systems such as Cousin [5,6] and Sara [3], which are integral help systems and which involve rewriting parts of the command language interpreter. Such integral help systems can provide detailed information on how to use a given utility, but they require that the user know which utility he needs, and thus violate the first condition listed above.

Yucca is designed to aid in the use of basic commands; it is not intended to be a command language version of the Programmer's Apprentice [8]. The type of answers which it supplies is limited to those which can be accomplished using a single command, or a simple sequence (UNIX piping) of commands.

Yucca is an outgrowth of the earlier system UCC, which was reported in [2]. It is currently being implemented in Franz Lisp for BSD 4.2 UNIX running on VAX[c] hardware. However, the design also encompasses basic concepts from other operating systems, such as TOPS-20, so that users switching from such systems can have their questions answered intelligently.

2. Design Rationale

The behavior of operating system commands is completely and formally describable. Once a query is understood, the problem is one of finding a correct answer; there is no concept of degree of correctness. To exploit this structure of knowledge, we have divided the system into two distinct but interconnected modules. The *knowledge base* provides a completely formal representation of the behavior of the operating system. It receives input in the form of formal queries in the specially designed query language *Quelya* and produces output which binds the variables of the query. The interface of the user to this knowledge base is provided by a separate module, the *natural-language front end*. The role of this module is to translate natural-language queries into equivalent Quelya requests and to convert bindings of these requests into natural-language answers. Thus, its purpose is similar to that of the natural-language components of database systems such as Planes [7] and Team [4], although the nature of the formal queries makes its design quite different. We note that this philosophy is distinctly different from that used in the UNIX consultant UC [9], in which the natural-language module is deeply involved in knowledge representation.

In this paper, the overall architecture of the knowledge base is outlined. Details of the natural-language front end will be discussed in a separate report.

* This work was performed in part under the auspices of the United States Department of Energy.

(a) UNIX is a trademark of Bell Laboratories.

(b) TOPS-20 is a trademark of Digital Equipment Corporation.

(c) VAX is a trademark of Digital Equipment Corporation.

3. The Nature of Formal Queries

In a conventional (nonprocedural) relational database query language, answering queries corresponds to the process of binding free variables in a first-order formula. For example, retrieving the set of all employees who make at least $30000/year may be expressed as

$$\{ x \mid \text{employee}(x) \wedge (\text{salary}(x) \geq 30000) \} .$$

In Quelya, there are two basic query formats, static and dynamic. *Static queries* are analogous to conventional database queries and are used to retrieve simple definitions. For example, the natural-language query "What is a pipe?" would be represented in Quelya as

```
(Find x ( x = description("pipe"))),
```

which is the Lisp-like Quelya notation for

$$\{ x \mid x = \text{description}(\text{"pipe"})) \}.$$

Just as static queries make use of a first-order static logic, *dynamic queries* make use of a first-order dynamic logic. Statements in such a logic follow the general flavor of Hoare-style propositional semantics [1]. The query "How do I list the contents of a file with control characters displayed?" is represented in Quelya as

```
(Find (P1 F1 Q1)
   (Dyn (P F Q)
      (Define P
         ((Clauses
             (Existsfile(#X))
             (type(file(#X)) = "plain")
             (Addnecessary P1))))
      (Define F
         ((transform F1))
      (Define Q
         ((Clauses
             (Contents(file(standard-output(%user)))
                   := contents(file(#X))
             (visible-nonprinting-characters
                   (contents(file(standard-output
                       (%user)))) = "yes")
             (Addimplications Q1)))).
```

Here P represents the *precondition* of the action, F the action, and Q the postcondition of the action. This is represented in more familiar notation as {P}F{Q}. The string #X designates a generic file path, while %user represents the terminal connection of the current user. P1, F1 and Q1 are the variables to be bound in the query. F1 names the action and is the primary variable to be bound. It will be bound to "cat -v" for the above example. P1 is bound to any *secondary preconditions*, in this case, to a statement that the file of #X must be readable and the standard output device of the user must be writeable. Similarly, Q1 to be is bound to any *secondary postconditions*. The assignment ":=" in the postconditions relates values before the action to those after; it can be formally eliminated by introducing new variables assigned in the preconditions.

This format may be used to express a wide variety of dynamic query types. However, at the present time, Yucca is only equipped to answer dynamic queries of the "How do I..?" variety, in which the action is the primary item to be bound.

4. Fundamental Components of the Knowledge Base

The knowledge base consists of two components, the encyclopedia and the formal semantics module. The *encyclopedia* is comprised of simple English language paragraphs describing key items, such as pipes and the overall file structure. It is used in a completely straightforward fashion to answer static queries and will not be further discussed in this paper.

The *formal semantics module* is comprised of the object and type definitions, the static predicates, the command semantics, and the item templates. The *object definitions* form the central core of the knowledge base. Objects currently modelled include files, directories, devices, terminal connections, and users. Each object definition consists of a hierarchically organized set of attributes. Part of the definition for the object type *file* is given below.

```
Intrinsic attributes:
     node-definitions [set-of(file-node)]
     owner [user-id]
     mode:
          protection [protection-type]
          type [file-type]
     contents [sequence-of(byte)]

Extrinsic subattributes of contents:
               (condition (type = "plain"
                                or type = "device"))
     numbered-lines [linetype]
     visible-nonprinting-characters [yes-no]
     end-lines-marked [yes-no]
     tabs-marked [yes-no]
     single-spaces [yes-no]
```

Intrinsic attributes are those which any instance of the object must have at all times (whether or not the value is actually known). *Extrinsic attributes*, on the other hand, can only have values when an explicit action has taken place. The term in square brackets next to each attribute is the *type* of the attribute; each such type is declared as a data type in the knowledge base.

The *static predicates* are used to describe states of instances of the various objects. For example, there is a static predicate *Readable*, which takes two arguments, a path definition and a terminal connection. Readable(P,c) is true if and only if the user at terminal connection c has read privileges on the file defined by path P.

Command semantics are represented propositionally. As an example, the semantic description of the basic (one file) UNIX cat command is given below.

```
Operation: HL-cat(t1,t2)
           t1 = path
           t2 = terminal-connection

Preconditions:
     AND : ¦ Readable(t1,t2)
           ¦ Writeable(standard-output(t2),t2)
```

```
Postconditions:
     AND : ; contents(file(standard-output(t2)))
                              := contents(file(t1))
           ; All "yes-no" attributes of
              contents(file(standard-output(t2)))
                                     = "no"
           ; numbered-lines(contents(file
                  (standard-output(t2)))) = "none"
           ; OK-cat-mods = "yes"
           ; Avail-cat-options-list = Cat-parameters
           ; Used-cat-options-list = ∅
```

Note that all extrinsic attributes of contents are initially set to "null" values. To obtain additional attributes of the target file, *filters* for the cat command are logically appended to the basic action. A description of the cat-visible-nonprinting-characters filter ("-v" option) is given below.

```
Preconditions:
     AND : ; OK-cat-mods = "yes"
           ; Member
                 ("visible-nonprinting-characters",
                             Avail-cat-options-list)
Postconditions:
     AND : ; Avail-cat-options-list
                := Avail-cat-options-list \
                {"visible-nonprinting-characters"}
           ; visible-nonprinting-characters
              (contents(file(standard-output(t2))))
                                           = "yes"
           ; used-cat-options-list
                := used-cat-options-list ∪
                {"visible-nonprinting-characters"}
```

The first precondition makes sure that it is all right to append filters pertaining to the cat command, while the second removes the applied feature from the available options list, to make sure that it is not used twice. By properly adding and deleting from the avail- and used- options lists of commands, complex option interaction has been modelled.

The powerful UNIX feature of input and output redirection is modelled in a similar fashion, via filters which change the values of standard input and standard output.

The description of the cat command contains all of the information needed to answer the formal query presented in the previous section; once P1 is bound to Readable(#X,%user)∧Writeable(standard-output(%user), %user), all of the postconditions are satisfied by appending the above filter to the basic cat command and defining F1 to be this composition.

For each attribute of each object, there is an *item template* which indexes commands pertinent to that attribute. In our example, the item template of interest is that which corresponds to the visible-nonprinting-characters attribute of the file object type. This template links this object attribute to the visible-nonprinting-characters filter of the cat command. Thus, the *solver module* of the knowledge base can, upon examining the postconditions of the query, quickly zero in on the appropriate command and filter(s).

A key feature of the object descriptions is the inclusion of attributes which are applicable to other systems, but inapplicable to the extant system. For example, the attribute *online-expiration-time* is included in the instrinsic attributes of file. This attribute, meaningless in UNIX but central in TOPS-20, is connected to an item template which contains an explanation of its inapplicability. Thus, the query "How do I determine the online expiration date of a file?" is answered with a statement indicating the inapplicability of this concept, rather than a statement of the form "Question not understood."

References

[1] Alagić, S., and M. A. Arbib, *The Design of Well-Structured and Correct Programs*, Springer-Verlag, (1978), 292 pp.

[2] Douglass, R. J., and S. J. Hegner, "An expert consultant for the UNIX operating system: bridging the gap between the user and command language semantics," *Proc. 1982 CSCSI/SCIEO Conference, Saskatoon, May, 1982.*

[3] Fenschel, R. S., and G. Estrin, "Self-describing systems using integral help," *IEEE Trans. Systems, Man, and Cybernetics*, (March/April, 1982), pp. 162-167.

[4] Grosz, B. J., "TEAM, a transportable natural language interface system," *Proc. 1983 Conf. on Applied Natural Language Processing*, pp. 39-45.

[5] Hayes, P. J., "Uniform help facilities for a cooperative user interface," *Proc. 1982 NCC, Houston.*

[6] Hayes, P. J., "Cooperative Command Interaction through the Cousin System," *Proc. Intl. Conf. Man/Machine Systems, U. of Manchester Institute of Science and Technology, July, 1982.*

[7] Waltz, D., "An English language question answering system for a large relational database," *Comm. ACM*, (July, 1978), pp. 526-539.

[8] Waters, R. C., "The Programmer's Apprentice: knowledge based program editing," *IEEE Trans. Software Engineering*, (January, 1982), pp. 1-12.

[9] Wilensky, R., "Talking to UNIX in English: an overview of UC," *Proc. 1982 AAAI Conf.*, pp. 107-110.

STEPS TOWARDS A THEORY OF EXCEPTIONS

James P. Delgrande

Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4

Abstract

This paper explores the notion of exceptions to general statements. In particular, an account is provided for those sentences that are in some sense universal, yet allow exceptions. Following from this account, it is shown how reasoning with such sentences may be carried out using existent techniques. Finally the problem of when to admit an exception, as opposed to when to abandon a general statement, is briefly discussed.

Introduction

Broadly speaking, science is concerned with uncovering the fundamental nature of natural phenomena and, in large part, deals with the discovery of general or universal laws about such phenomena. In a word, science *systematises*. Such laws, of course are never known to be true and, moreover, are rarely known to not be exceptionless. Rather, one applies a law in conjunction with an indefinitely large collection of "additional assumptions". Thus "water boils at 100°C" will be true of a particular sample of water only if the water is pure, the pressure is 760 mm., the measuring device is accurate, etc. If a sample fails to boil at 100°C then, one can always claim that some additional assumption has been violated, even if such a statement is not known. Thus a law may be retained even though (logically) falsified.

The overall direction of this paper derives from these intuitions. It is assumed that concepts (including natural kind terms) do indeed have underlying necessary and sufficient conditions, but that such conditions are essentially unknowable. The aim of a learning system in this instance is to hypothesise a theory of such terms. Now, as indicated above, many statements that can be made about such concepts are not exceptionless and thus cannot be necessarily true of a class. The first part of the problem then is to precisely specify what we mean by a "universal" statement which allows exceptions. Note though that we specifically reject a default interpretation of such statements. Thus we would not interpret a statement such as "all elephants have four legs" by "an elephant typically has four legs". Before proceeding though, we review the notion of exception and the allied notion of default.

Exceptions and Defaults

An *exception* is basically a counter-example to a general statement or rule. Thus if Clyde were a three-legged elephant (to pick a perfectly random example), he would be an exception to the rule of four-leggedness for elephants. A *default* on the other hand is a rule that allows an inference to be drawn in the absence of contrary information. Thus if one knew only that Clyde was an elephant, one might assume that he had four legs. Exceptions then arise when formulating (or verifying) a theory of a domain; defaults arise when reasoning within such a theory. Hence exceptions arise with respect to the *meaning* of a term; defaults with respect to a *description*. The most fully developed approaches to default reasoning are given in [5] and [7].

An informal division of exception types into individual exceptions and exception classes is given in [4]. The first type includes, for example, a raven that has been painted red, or an elephant with three legs as the result of an unfortunate accident. The second type can be divided into two subtypes. First there are non-subclass forming exception classes, such as albino animals; and second there are subclass forming exception classes. Penguins are an example of this latter type since they are flightless, in contrast to birds which, on the whole, are taken to fly. Again we emphasize that we do not want a default interpretation of such statements. Rather we want to say something to the effect that it follows (in some sense) from the nature of birdhood that birds fly; but it follows from the nature of penguinhood that they don't fly, even though penguins are, of course, birds.

We have assumed that necessary and sufficient conditions for kinds, such as ravens or water, exist but aren't knowable. (See [9] for a collection of papers on this subject.) Thus a naive theory of water states that it freezes at 0°C, is transparent when liquid, etc. Such statements are clearly potentially falsifiable. However, if such a statement is falsified, it is not perforce rejected, since it may be part of the best available theory of water, and does provide useful information about the substance.

A less naive theory states that water is H_2O, and, following [6], we can say that *if* water is H_2O, it is *necessarily* H_2O. In this case the definition of water admits no exceptions; the hydroxyl radical OH, for example, is simply not water. In addition, our naive theory of water can now be explained in terms of this more sophisticated theory, as can the exceptions. Lastly, in the presence of a full chemical theory, water can be discharged as a primitive term and ***defined*** in terms of more fundamental concepts.

A Semantics for Exceptions

We associate with each predicate P in our theory two sets, $\mathbf{E}_P$ and $\mathbf{S}_P$. $\mathbf{E}_P$ contains those conditions hypothesised to be necessary for P-hood, and is thus the set of (hypothesised) ***essential*** features. This set is hypothesised to be a subset of the actual (unknowable) necessary and sufficient conditions for P-hood, $\mathbf{U}_P$. We will write $\mathbf{U}_P(x)$ to mean that x satisfies these conditions. The set $\mathbf{S}_P$, of ***significant*** statements contains those features that are true (later we will claim necessarily true) of P, all things being equal. Thus a sentence of $\mathbf{E}_P$ can have no exceptions while a sentence of $\mathbf{S}_P$ can. Both though may be revised. We require $\mathbf{S}_P$ because in naive, or less than physical theory, $\mathbf{E}_P$ may [illegible] exceptions. Thus, for example, for a theory of birds (B), including ravens (R) and penguins (P), we may have:

$$\mathbf{E}_B = \{\, is\_animal\ ,\ \cdots \}$$
$$\mathbf{S}_B = \{\, unif\_colouring\ ,\ flies\ ,\ 2\ wings\ ,\ \cdots \}$$
$$\mathbf{E}_R = \{\, is\_bird\ ,\ \cdots \}$$
$$\mathbf{S}_R = \{\, black\ ,\ flies\ ,\ 2\ wings\ ,\ \cdots \}$$
$$\mathbf{E}_P = \{\, is\_bird\ ,\ \cdots \}$$
$$\mathbf{S}_P = \{\, black\_\&\_white\ ,\ \neg flies\ ,\ 2\ wings\ ,\ \cdots \}$$

The union of all hypothesised essential statements will be denoted $\mathbf{E}$, and significant statements will be denoted $\mathbf{S}$. Both $\mathbf{E}$ and $\mathbf{S}$ are assumed to be finite.

Now, since $B \supset R$, we must have that $\mathbf{U}_B \subset \mathbf{U}_R$, and thus can write $\mathbf{U}_R = \mathbf{U}_B \cup \mathbf{U}_R'$, where $\mathbf{U}_R'$ is specific to ravens. We must also have $\mathbf{E}_B \subset \mathbf{E}_R$, and thus $\mathbf{E}_R = \mathbf{E}_B \cup \mathbf{E}_R'$, where $\mathbf{E}_R'$ is again specific to ravens. Also if

$$\mathbf{E}_B = \{p_1, \cdots, p_n\}$$

then we can write

$$\mathbf{E}_R = \{p_1, \cdots, p_n, q_1, \cdots, q_m\}$$

where each of the q_i represents a new condition specific to R, or represents a restriction among the p_i. Containment relations, and hence inheritance (if such is defined) must clearly be strict. Thus, either way, a raven being a bird necessitates that it also be an animal.

It is easy enough to show that these hypothesised essential conditions interact reasonably. In fact, the set of $\mathbf{E}_i$ under containment form an upper semilattice, which has been claimed to be an appropriate structure for taxonomies of natural kinds [3].

In the case of the significant properties, things clearly are different. Consider the example of $B \supset P$ and $B \supset R$. The set $\mathbf{S}_B$ follows (in a sense to be specified) from $\mathbf{U}_B$, while $\mathbf{S}_R$ follows from $\mathbf{U}_R$. If $\alpha \in \mathbf{S}_B$, there are three cases:

i) $\alpha \in \mathbf{S}_P$ - Here we could implement standard ("default") inheritance. Thus birds have feathers and penguins have feathers.

ii) $\neg\alpha \in \mathbf{S}_P$ - Then $\neg\alpha$ "follows from" $\mathbf{U}_P$. But $\mathbf{U}_P = \mathbf{U}_B \cup \mathbf{U}_P'$, and $\alpha \in \mathbf{S}_B$. Thus $\neg\alpha$ follows, at least in part, from the meaning of penguin and, hence, serves to distinguish its meaning from the inclusive set of birds. Thus birds fly, but penguins don't.

iii) $\alpha \notin \mathbf{S}_P$ - This can be treated as in i). To use a different example, birds fly, chickens may.

This then fixes what is meant by a universal statement that admits exceptions: it is a statement that follows (all things being equal) from the necessary conditions of that class. This means that such a statement, all things being equal, is ***necessarily true***. Thus an elephant, all things being equal, necessarily has four legs.

The notion of "all things being equal" clearly corresponds to the "additional assumptions" for the predictions of scientific theories, discussed in the introduction. In reasoning about an individual, "all things being equal" is equivalent to "in the absence of conflicting information", which of course is the standard interpretation of a default rule.

Thus if x is a member of a particular class, B, we get:

$$B(x) \equiv \mathbf{U}_B(x)$$

and can write:

$$(\mathbf{U}_B(x) \wedge \neg K \neg \alpha(x)) \supset H\alpha(x) \qquad\qquad \alpha \in \mathbf{S}_B\,.$$

$K\alpha$ ($H\alpha$) is interpreted as "α is known (resp. hypothesised) to be true". Thus, if $B(x)$, x necessarily satisfies the (unknown) conditions, $\mathbf{U}_B$. However, as a result, if it is not known to not satisfy a property in $\mathbf{S}_B$, then we can hypothesise that it does. The use of K and H is covered in [1], which provides a semantic and proof-theoretic analysis of first-order predicate calculus augmented by these operators. The resultant language, HKL in turn is a straight-forward extension of KL developed in [5] (which also provides a general account of the above default reasoning).

Now, within a theory of a domain, all sentences of the sets $\mathbf{E}$ and $\mathbf{S}$ are hypothesised to be necessary (either strictly, or in a default sense). However these sentences can be transformed so that only predicate letters appear within the scope of the necessity operator, and so that all instances of the necessity operator have only predicate letters (i.e. no connectives or quantifiers) in their scope. This means that we can use the machinery of HKL for representing and reasoning about the sets $\mathbf{E}_p$ and $\mathbf{S}_p$, where the sentences of $\mathbf{S}_p$ are represented as default rules, as indicated above.

Exception Types

Earlier we gave an informal breakdown of exception types, viz:

i) Individual exceptions

ii) Exception classes

a) non-subclass forming

b) subclass forming

Now, exception conditions may be considered as predicates and hence, intensionally, as classes; and thus should be amenable to the preceding analysis.

To begin with, individual exceptions appear to correspond to the situation where the reason for the exception is known or assumed knowable. Thus, if a raven has been painted red, then the reasons behind that particular exception are fully specified. Similarly, if one encounters a green raven, then the claim can always be advanced that one could discover (not hypothesise) the cause if one tried hard enough; and thus that the reason is potentially knowable. This is in contrast to albinohood where at best the underlying mechanism may be conjectured. The key point is that with individual exceptions the exception can (in fact or allegedly) be immediately discharged.

In the case of exception classes, for example albinohood (A), the set $\mathbf{E}_A$ is never known. $\mathbf{E}_A$ may be completely specified, and even true; and if true is necessarily true; but is never known to be true. Thus such exceptions are kinds in exactly the same manner as are ***water*** or ***raven***. An exception class then may be discharged only when the class to which it applies is definable in more fundamental terms. Thus once we can propose a definition for the term "animal" (in terms of DNA or whatever), we can also fully define albinohood. Similarly a full specification of birdhood and penguinhood leads to a full explanation (specification of $\mathbf{E}_B$ and $\mathbf{E}_P$) as to why the former flies (all things even now remaining equal) and the latter does not.

Pragmatic Considerations

To this point we have discussed the meaning of general statements that allow exceptions, and in what sense they follow from a set of necessary conditions. However a number of problems, largely of a pragmatic nature, remain. These problems likely cannot be answered without considering the structure and use of the underlying reasoning system and without including the *a priori* beliefs held concerning the domain.

First there is the distinction between the sets **E** and **S**. Certainly a sentence which has exceptions cannot be a member of **E**. On the other hand, the assertion that water is H_2O belongs in **E**. However grey areas remain. For example, because an albino penguin has never been observed is no reason to believe that none (can) exist. Such a contention may be strengthened by the existence of albinos in similar kinds (such as ravens and ducks) or perhaps in more general kinds (such as mammals). However, before anything can be accomplished, notions such as "strengthen" and "similar", used in the preceding sentence, must be defined.

The major problem though concerns automating the decision of when an exception should be admitted and when a general statement should be modified or abandoned. This however presupposes that "coherent" general statements can be formed; there is certainly no difficulty in constructing arbitrary sentences that imply any given set of data. This consideration however leads directly to the problems of induction and confirmation; problems which are, in the general case, inordinately difficult (see [2] or [8]).

If we assume that the general statments are indeed reasonable, then the only answer appears to be that violated statements are retained if they are credible and useful. By credible we mean "may be necessarily true, all things remaining equal". However this can be judged only on the basis of *a priori* beliefs about the domain, and the contents of the knowledge base.

It is difficult also to specify what is meant by a statement being "useful". In particular, a statement would be useful when it could be used by the reasoning component of the knowledge base. However there is also the role that the statement plays in the theory of the domain. Thus, for example, the conjecture "all ravens are black" may provide evidence for higher-level conjectures such as "all bird species are coloured uniformly" or "all animal species' colouring varys uniformly", as well as related conjectures, such as "swans are white", and thus perhaps "bird species have similar characteristics". To abandon this first conjecture may then involve abandoning a host of related conjectures. Thus exceptions allow the overall structure of a knowledge base to be maintained. The problem of when to admit exceptions then reduces in part to the problem of knowledge base systematisation.

Conclusion

This paper has discussed the problem of exceptions to general statements. The question of the meaning of exception-allowing sentences has been approached by distinguishing two types of conjectures concerning a class. The first corresponds to hypothesised necessary conditions. The second corresponds to default necessary conditions; these are analogous to the "additional assumptions" of scientific theories. In the case of natural kind terms, the first set is assumed to likely be virtually empty; the aim here however has not been to show how such such essential conditions can be found, but rather to explicate the semantics of exceptionless and exception-allowing statements. A second problem, of reasoning with these statements, follows from an easy adaptation of existing default theories of reasoning. A third problem, that of when to admit exceptions, is asserted to depend on the use and overall structure of the theory and underlying knowledge base.

This approach treats classes uniformly. While the discussion has been geared toward natural kind concepts throughout this paper, the approach could equally well be applied to strictly definitional terms -- for example learning about kinship relations. In addition, exception classes themselves may be treated within this framework.

References

[1] J.P. Delgrande, Ph.D. thesis (in preparation), Department of Computer Science, University of Toronto, 1984

[2] N. Goodman, *Fact, Fiction and Forecast*, 3rd ed.,

Hackett Publishing Co., 1979

[3] D.J. Israel, "On Interpreting Semantic Network Formalisms", Technical Report 5117, Bolt Beranek and Newman Inc., Sept 1982

[4] Y. Lesperance, "Handling Exceptional Conditions in PSN", *Proc. CSCSI-80*, 1980, pp 63-70

[5] H.J. Levesque, "A Formal Treatment of Incomplete Knowledge Bases", Ph.D. thesis, Department of Computer Science, University of Toronto, 1981

[6] H. Putnam, "The Meaning of 'Meaning'" in *Mind, Language and Reality: Philosophical Papers Volume II*, Cambridge University Press, 1975, pp 215-271

[7] R. Reiter, "On Reasoning by Default" *TINLAP-2*, 1978, pp 210-18

[8] I. Scheffler, *The Anatomy of Inquiry: Philosophical Studies in the Theory of Science*, Hackett Publishing Co., 1981

[9] S.P. Schwartz (ed.), *Naming, Necessity and Natural Kinds*, Cornell University Press, 1977

Canadian Artificial Intelligence in the Next Decade

Funding, Strategies, and the Role of the CSCSI/SCEIO

Gordon McCalla

Vice Chairman, CSCSI/SCEIO
Department of Computational Science
University of Saskatchewan
Saskatoon, Saskatchewan, Canada S7N 0W0

Nick Cercone

Chairman, CSCSI/SCEIO
Department of Computing Science
Simon Fraser University
Burnaby, British Columbia, Canada V5A 1S6

Abstract

In this paper we outline the role of the Canadian Society for Computational Studies of Intelligence/Societe Canadienne pour Etudes d'Intelligence par Ordinateur (CSCSI/SCEIO) in the history of Canadian artificial intelligence (AI). We also discuss possible future directions for AI in this country, based on the opinions of the membership of the CSCSI/SCEIO as expressed in a survey carried out in 1983. Finally, we pose several questions about the role of the CSCSI/SCEIO in Canada in the coming years.

1. Introduction

The CSCSI/SCEIO is now ten years old and the time is appropriate for an examination of its past and future roles in Canadian AI. This is doubly important given the recent upsurge of interest in artificial intelligence shown by other computer scientists, government funding agencies, private funding agencies, and the media. The CSCSI/SCEIO has done a remarkable amount in its decade of existence, especially considering the volunteer nature of the organisation, and it has the potential to be even more useful and relevant in the next decade.

In this paper we examine the past triumphs of the CSCSI/SCEIO (section 2). We then present the views and feelings of the CSCSI/SCEIO membership about future directions for AI in this country based on the responses given to the recently distributed questionnaire (section 3). Finally (section 4), we focus in on what we feel are the most important decisions facing the CSCSI/SCEIO at the moment and leave some 'open' questions in order to stimulate discussion among the attendees at this conference and generate reaction from the CSCSI/SCEIO membership in general.

2. The CSCSI/SCEIO

The CSCSI/SCEIO was founded in 1973 to act as the umbrella organisation for AI in Canada. Deliberately circumlocutious, the name was chosen in order to attract a wide range of research interests, including AI, but also encompassing areas like image processing, pattern recognition, cognitive psychology, computational linguistics, man-machine studies, and so on. Pattern recognition and image processing now have their own organisation (CIPPRS), as do graphics and man-machine studies (CMCCS/ACCHO), but the CSCSI/SCEIO still continues to attract some afficiandos of these areas as well as people with "mainstream" AI interests.

Originally an informal grouping of about 30 researchers, mainly academics, the CSCSI/SCEIO is now nearly 300 strong. The core of the organisation is still the academic AI research community (and, in fact, its executive officers have always come from this community), but there are now members from government, research laboratories, and the private sector. The organisation, while still maintaining much of its informal character, has been legally constituted (receiving its "letters patent" under Ontario law in 1980). It is also now a special interest group of the Canadian Information Processing Society (CIPS).

The society has accumulated a number of functions over the years. Its first major activity was to distribute a newsletter. Compiled by the UBC AI group, the first one of these appeared shortly after the formation of the society in 1974. Typical of the early newsletters, it was considered to be a "one-shot" enterprise and involved significant effort, not to be soon repeated. This experience resulted in responsibility sharing from one AI group to another around Canada to produce approximately yearly newsletters. Thus, the second newsletter was produced by University of Western Ontario, the third at University of Toronto, the fourth at University of Alberta, and the last one of the early series at University of Ottawa. At this point the membership decided that the newsletter should appear more frequently and with a lot less effort than the special case newsletters that had been produced to that time. Under Wayne Davis' co-ordinating editorship, a joint newsletter (shared among CSCSI/SCEIO, CMCCS/ACCHO, and CIPPRS) was formed, with sub-editors representing each society. The lofty goal was to produce a quarterly newsletter; in actuality, only half that number has been forthcoming, but perhaps the frequency will pick up as the popularity of AI grows and the number of submissions consequently increases.

Another major activity of the CSCSI/SCEIO has been to host national AI conferences, held in even numbered years. The precursor to these conferences was an AI workshop held at University of Ottawa in 1975 under John Mylopoulos' chairmanship. The first real conference (with refereed extended abstracts) was held at UBC in August 1976, organised by Richard Rosenberg and Alan Mackworth. It attracted around 60 people (this is remarkable considering that the decision to hold the conference preceded the conference by only 6 months). A full proceedings was produced, as at all future conferences, and a marvellous salmon barbeque was held at Richard Rosenberg's mansion. The first conference established the tradition of excellence in invited speakers with Zenon Pylyshyn's keynote address.

The second conference was held in 1978 at University of

Toronto, chaired by Ray Perrault with Ted Elcock as programme chairman. Held in muggy July weather, it attracted 120 people. Despite a less than outstanding Chinese banquet, the conference still mantained the relaxed friendly atmosphere which has become a CSCSI/SCEIO conference tradition. The keynote speaker at the second conference was Alan Mackworth.

The third conference was held in May 1980 at University of Victoria under the direction of the Alberta's AI establishment (Wayne Davis, general chairman; Len Schubert, programme chairman). It was notable not only for its large attendance (around 150 people), but also for being the first conference held in conjunction with the CIPS national conference and for its large number of invited speakers (Ted Shortliffe, Ray Reiter, Jerry Hobbs, Steve Zucker, Hans Berliner, Robert Wilensky). The 1980 conference was also the first to compete with a AAAI conference (whose initial conference was held in August 1980). Up until that time, the CSCSI/SCEIO had hosted the only North American AI conference in off-IJCAI years, although there had been a couple of "special interest" TINLAP workshops.

The fourth conference was held at University of Saskatchewan in May 1982, also in conjunction with the CIPS conference. Gord McCalla was the general chairman; Nick Cercone the programme chairman (the first collaboration of the dynamic duo). Considering that IJCAI had been held in Canada only 9 months before, and that AAAI now had their conference organisation in high year, attendance dropped to approximately 80 people. The number of invited speakers remained at six (Roger Schank, Ron Brachman, B. Chandrasekaran, Scott Fahlman, Bonnie Lynn Webber, and Robert Woodham), the size of the programme committee doubled and the convivial atmosphere was maintained, including a great banquet at the Faculty club.

The fifth conference is being held at Western Ontario with Ted Elcock as general chairman and John Tsotsos in the programme chair. This conference is the first to require full papers, rather than extended abstracts, which should further enhance the quality of the papers. This conference also marks the re-establishment of independence from CIPS, due to the locale problems posed by the fact that CIPS is an annual conference and also due to the minimal cross-fertilization that has occurred between the two conferences in the past.

Before leaving the conference beat, it should be mentioned that the CSCSI/SCEIO was the host organisation for the International Joint Conference on Artificial Intelligence (IJCAI) in 1981 when it was held in Vancouver. Richard Rosenberg had the herculean task of handling local arrangements. He wisely decided not to have the salmon barbeque at his house. The conference was a great success, marred only slightly by a bus strike (providing authentic local colour at least).

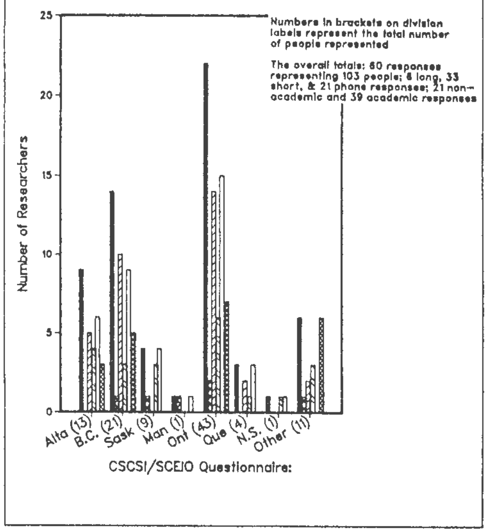
The CSCSI/SCEIO is also involved in other activities. Currently a proposal has been put forward for the society to sponsor an international AI journal "Computational Studies of Intelligence". The executive is currently engaged in negotiations with a publisher with publication planned to commence in 1985. The organisation, through its executive, has also produced a document entitled "Directions for Artificial Intelligence in Canada" [1] which compiles the results of a survey of its membership on current AI research activities and future research directions.

In section 3 we single out the "directions" component of this document for special attention. The section is more or less reproduced exactly from [1]. Then, in section 4 we pose several important questions which should be answered about the CSCSI/SCEIO's role in Canadian AI. We hope these will stimulate discussion at this meeting and beyond.

3. Directions for Canadian AI

We summarise the results of a questionnaire sent to the CSCSI/SCEIO membership in October, 1983. This questionnaire is a short, multiple choice version of a longer "subjective" questionnaire sent out in April of 1983. We received 60 responses to the questionnaire representing 103 individuals, categorised as follows: 6 responses from the long questionnaire and 54 responses from the short questionnaire. Of these 54 responses, 33 were mailed in and 21 were the result of a follow-up telephone campaign. The breakdown of the responses geographically is shown in graph 1.

GRAPH 1 – Summary of Responses



We integrated into one "lump" the responses from the long questionnaire and the written and phone responses to the short questionnaire. We then sub-divided this lump into academics residing within Canada (dubbed "academics") and all other people (dubbed "non-academics"). Non-academics mostly consist of individuals in private industry within Canada, but also include personnel of government laboratories, private research laboratories, and people residing in the United States or overseas. It was felt that the vast majority of Canadian AI research is being undertaken in the universities and we wanted to single out this group to see if they held any substantially different views from the rest. The division is fairly arbitrary, and the distinctions in viewpoint between the two groups have turned out to be fairly subtle (except where explicitly indicated below), but it seemed important nevertheless to keep the data separated in this explication.

We explain the results of the questionnaire below. Accompanying the verbal description of the results are graphs outlining the percentages of respondents picking a particular answer in each of three categories: academics, non-academics, and overall. The academic percentages were computed out of 39, the number of Canadian academics who answered the questionnaire; the non-academic percentages were computed out of 21, the number of other respondents; and the overall percentages were calculated out of 60, the total number of respondents to the questionnaire. Since respondents often made several choices to a question (or no choices), the percentages don't normally add up to 100 percent in any category. In the graphs, the overall percentages are represented by the solid bar on the left, the academic percentages by the cross-hatched bar in the middle, and the non-academic percentages by the striped bar on the right.

Also quoted at length below are remarks made on the questionnaires by respondents. We have attempted to pigeonhole the comments opposite the appropriate question on the questionnaire. No comment is quoted more than once, even if it has relevance to several points. Not every remark made on the questionnaires has been quoted. We hope we have done justice to the feelings of the respondents and haven't misrepresented anyone's opinions by quoting out of context or in the wrong place.

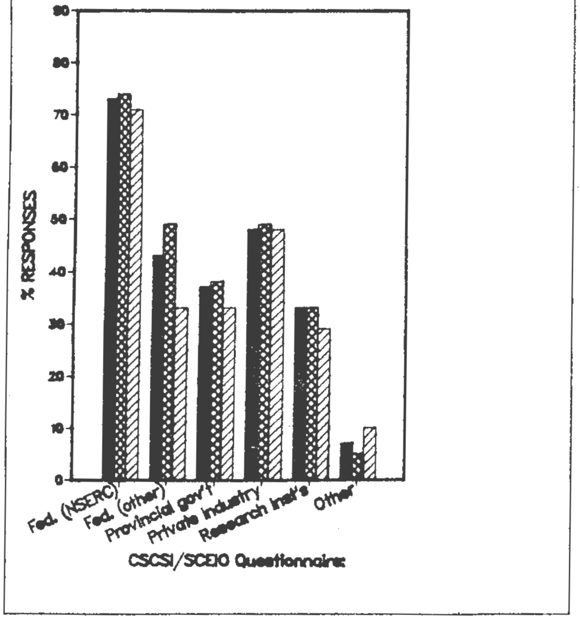
Part 2 – Financial Support

In part 2 of the questionnaire respondents were asked to comment on various aspects of funding for AI in Canada. Needless to say, most people felt that current levels of funding were inadequate, although 2 respondents didn't agree. As one of them put it: "more people are the first need - throwing money [at a problem without the people] wouldn't necessarily help much". Others found the question difficult to answer due to their lack of knowledge of the costs of doing AI, disinterest in the question, or a feeling that specific projects should be set first before discussing funding.

The findings of the survey about funding are outlined in graphs 2 through 5. They indicate that NSERC is the agency of choice for distributing funds, that an additional 10 to 100 million dollars should be allocated to AI over the next decade, that AI should be supported in conjunction with other areas, and that equipment is the major priority (followed closely by release time for current researchers, additional support staff, and extra researchers). Lets look at each question in more detail.

The first question in part 2 asked about who should provide funds for AI research in Canada (see graph 2). The fact that NSERC is looked on favourably (by academics as well as non-academics) is reassuring for current government funding policies. It reflects a realisation that there is a role for basic research and that only NSERC currently has that mandate. It was suggested that "NSERC should declare [AI] as one of the areas eligible for strategic grants", presumably explicitly rather than implicitly as part of computers and communications as is the case now.

GRAPH 2 – 2. Financial Support
Additional Funds: Provided by



Private industry was the second most favoured source of funds, again with remarkable agreement between academics and non-academics. One respondent thought that private industry should be made more aware of the practical aspects of AI. Federal agencies other than NSERC came in a close third; among the agencies mentioned were the Secretary of State's office, the Medical Research Council, the Department of Communications, and the Department of National Defence. One respondent expressed the sentiment that these other government departments "should fund applied research leaving basic research to NSERC". Provincial governments ranked fourth: the provincial science councils were mentioned, as were the universities. Research institutions as sources of funds for AI did not rank highly, an opinion that may be changing as organisations such as the Canadian Institute for Advanced Research (which is currently funding a project with a strong flavour of AI) prove that private capital does exist for research projects in the field.

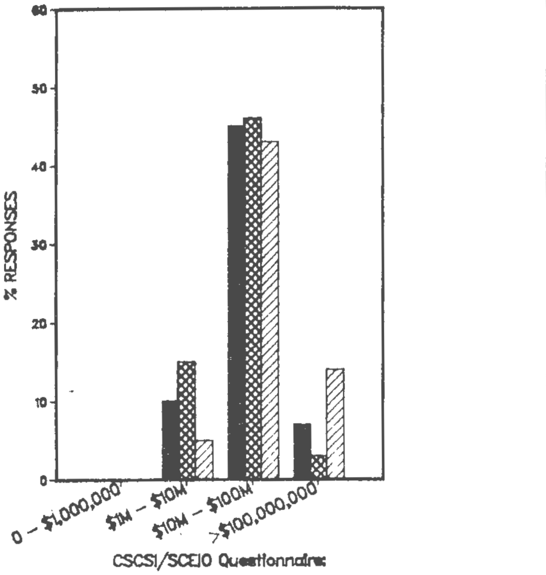
Perhaps the most interesting suggestions made about the source of funds came among the 7 percent "others". One person suggested that "the Japanese have called for collaboration on the Fifth generation project - Canada should investigate the offer of collaboration [rather than directly competing against them]". Another well thought out response said that "a granting agency is needed to fund AI software developments requiring $200,000 or more (for manpower only). Such developments should be targeted to areas of potential interest to industry, perhaps 6 or more such grants per year. The effect and aim would be to get industry interested while providing them with experienced people who could further develop the products. This in turn would increase the demand for graduates."

The second question in part 2 (how much money should be spent?) was sporadically answered (see graph 3). Most respondents felt that 10 to 100 million dollars in additional funding over the next 10 years (1 to 10 million per year) was appropriate, although one person said that "$10 million / year is peanuts". Whether even 1 million dollars per year is realistic is open to question in these days of restrained budgets, but the unanimous opinion was that this level of extra funds is the minimum required.

The third question in part 2 (should AI be funded in isolation or with other areas?) also had decisive results. There was less than a 100 percent response rate, but among the two-thirds who did respond, there was a 4 to 1 ratio in favour of funding AI in conjunction with other areas (see graph 4). Computer science was the most prominently mentioned "other" area, including VLSI design, hardware, software, CAD/CAM, evaluation of technology, simulation, distributed processing, and software engineering. Also mentioned were education and engineering as well as areas directly related to AI (or included in it) such as robotics, logic programming, pattern recognition, cognitive psychology, man-machine interfaces, psycho-linguistics, automatic programming, and computational linguistics. Comments supporting the need for funding AI in conjunction with other areas included "excursions into these 'esoteric' fields are not isolated ventures, but a continuum"; "[it is] impossible to define hard boundaries for AI"; and "the last thing AI wants to do is grow in a vacuum".

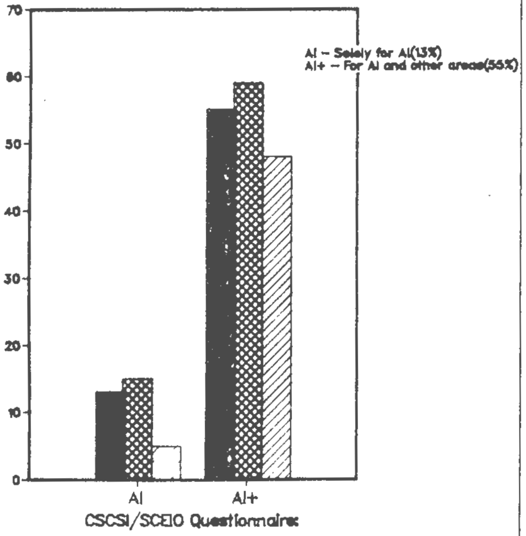
The biggest worry about supporting AI in conjunction with another area seemed to be a feeling that there had been prejudice against AI in the past (especially within computer science): "[AI must achieve] the removal of the prejudice often directed at AI by colleagues and federal and provincial funding agencies"; "more computing science support might have NO effect on AI support"; and "it should be guaranteed that

GRAPH 3 – 2. Financial Support
Additional Funds: How much



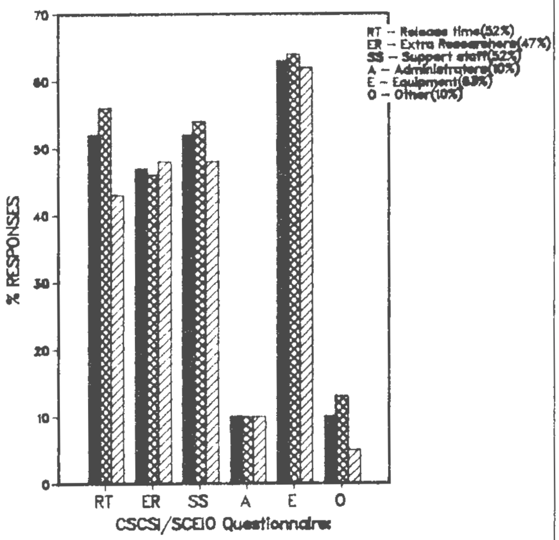
the increased support would be passed on to AI by the computer science funding agencies". To avoid this problem, it was suggested that it should be possible to "raise AI's profile within computer science so as to increase [its] visibility to NSERC". Perhaps the whole question will soon be moot, as the following humorous observation attests: "if the Japanese are right, AI will be computer science (or vice versa) by 1990".

GRAPH 4 – 2. Financial Support
Additional Funds: Allocation



The final question in part 2 of the questionnaire was an attempt to find out on what to spend any extra money (see graph 5). Surprisingly, equipment seemed to be the most in demand, which is good since extra equipment would seem to be easier to buy than extra people. But, there was a large demand for release time from current duties (especially among academics from teaching and administration), extra researchers, and support staff (often explicitly identified as programmers). Very few respondents were keen on the need for more adminstrators. Among the other needs mentioned were for maintenance and infrastructure support, graduate student support, software, travel, and a research centre for AI. The most eloquently expressed "other" need was to repatriate Canadians: "[we must win] Canada's best back from the U.S. by paying competitive salaries"; "graduate students sent abroad should be required to return for a reasonable period"; and "[we should] keep good young Canadians in Canada and convince expatriates to return (more money, better working conditions, AI research labs., etc.)". One respondent did note, however, that "there are actually a growing number of AI researchers in Canada despite the 'brain drain' - it is now 'de rigeur' to have one AIer even in a small department".

GRAPH 5 – 2. Financial Support
Additional Funds: Targeted towards



Part 3 – Strategies

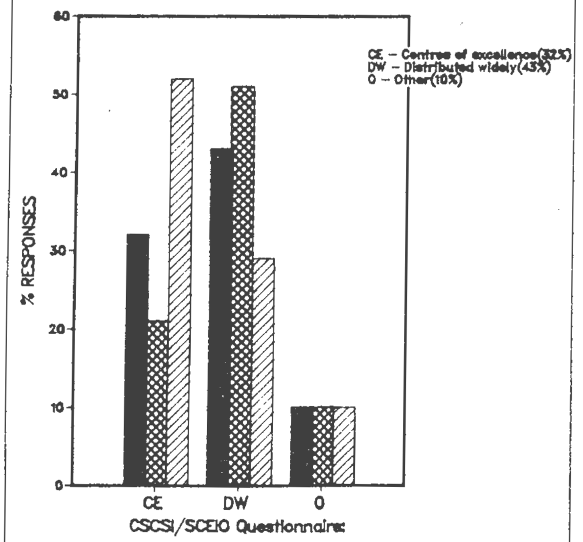
In part 3 of the questionnaire respondents were asked to comment on a number of different possible strategies for AI in the next decade. This part was more universally answered by the respondents than was part 2. It also led to a couple of disagreements between the academics and the non-academics.

Graphs 6 through 11 summarise the responses to these "strategic" questions. Here is a synopsis of the findings: AI research should be widely distributed geographically (although academics and non-academics differ on this), focussed on a wide variety of topics, directed towards both practical and theoretical results, carried out in universities, aimed at long term goals, and oriented towards software rather than hardware. Each question in part 3 generated an interesting variety of opinions.

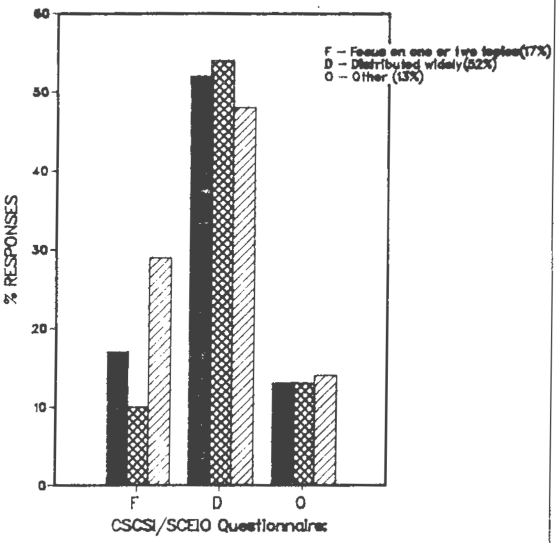
In the first question respondents were asked to contrast two essentially different geographic arrangements: centres of exellence vs widespread research (see graph 6). There is a substantial difference of opinion between academics and non-academics on this issue: non-academics were 2 to 1 in favour of concentrating research in centres of excellence (although sometimes the number of centres suggested was somewhat greater than 2), while academics were over 2 to 1 against such concentration, preferring instead a widely distributed research effort. The reasons for this difference of opinion are open to speculation: it could merely be an artifact of the particular subset of people who responded to the questionnaire, or it could represent a distinctly different approach favoured by the two camps. The fact that the majority of both academics and non-academics preferred the research focus to be wide ranging (see graph 7) further clouds the issue, since it would seem to be easier (although by no means essential) for a researcher to maintain a different perspective on the world if he or she isn't physically close to a critical mass of people with a particular world view.

This question of centralized vs widespread activity generated many comments. In favour of centres of excellence were remarks such as "one or two large AI centres is all the country can support". Against centres of excellence were comments such as "AI is too diverse and ill-formed yet to put all our eggs in a few baskets"; "centres are notoriously difficult to create"; and a plea for the ability to do AI research at smaller universities: "there are more graduate students, perhaps, at a big school, but often less freedom, less open-mindedness, less control, more anti-AI bias". Many respondents suggested letting research grow wherever it found root ("anarchism") or saw the need for a "judicious mix" of the two strategies: "both [centres and distributed] - we have to have at least one AI person in each ... university to teach the people who come to the centres to be AI graduate students"; "practical work, i.e. functioning AI systems, should be concentrated in centres of excellence; theoretical work can be distributed more widely".

GRAPH 6 — 3. Strategies: Geography



GRAPH 7 — 3. Strategies: Investigations

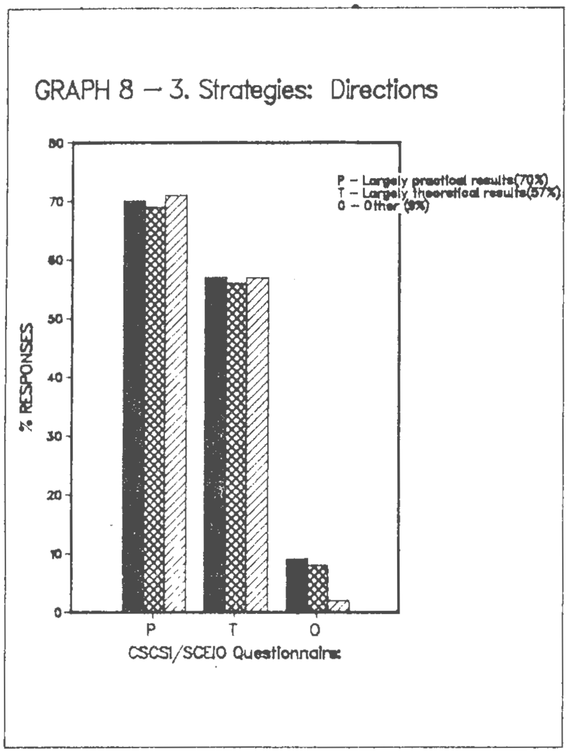


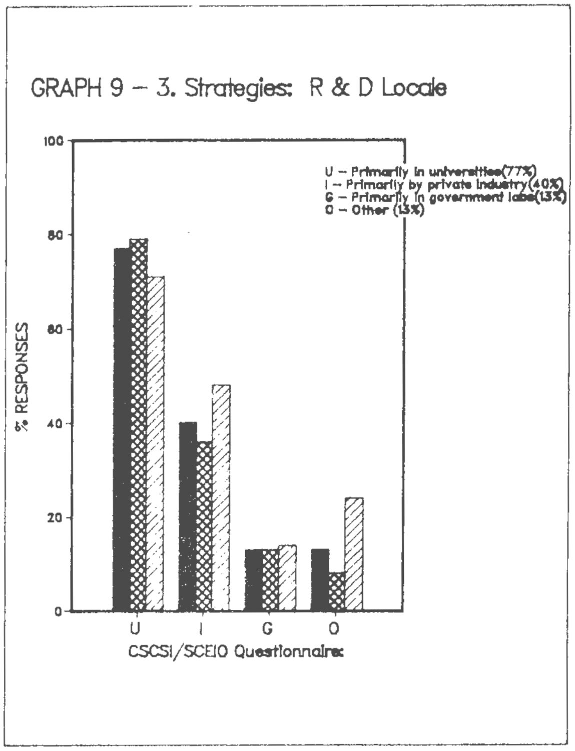
One frequently expressed desire was for electronic networks to allow geographically distributed researchers to keep in touch: "one thing that would make a big difference is if CSCSI could organise us all on a net". Another suggestion was funds to increase the mobility of people so they could "interact on a personal basis, [make] visits, [take] leaves, etc." This might allow a sort of distributed critical mass, and would certainly enhance communication, idea sharing, and the launching of co-operative projects among different institutions.

The second question in part 3 of the questionnaire asked about whether the focus of AI research should be on one or two topics or distributed more widely. There was considerable opinion (see graph 7) that research topics should be diverse since to ensure self-sufficiency Canada "must provide expertise in all areas of AI" and that the difficulty of predicting successful avenues of research in advance requires "broad based support". There was a strong feeling that it was impossible at any rate to easily direct what people chose to investigate, i.e. that the focus should be "research driven" and that we "don't want a lot of regulations and government direction".

Those who did suggest focussing on a couple of topics (and more non-academics thought this way than did academics) made general comments such as "priority [should be] given to existing strengths and national needs", or made more specific suggestions about particular topics: e.g. "Canada could concentrate on natural language (machine translation), [interpreting] satellite photos, and 2 or 3 other relevant topics"; or Canada should focus on "specific needs such as French/English translators, mining robots, and grain train schedulers". This topic can be concluded with the following two remarks: "those capable of spending the poor taxpayers' dollar are more influenced by Japan's 5th generation than by anything we can do"; and "on the whole [AI] looks like an interesting mosaic after an earthquake".

The third question in part 3 was aimed at sampling opinion on whether theoretical research or research oriented

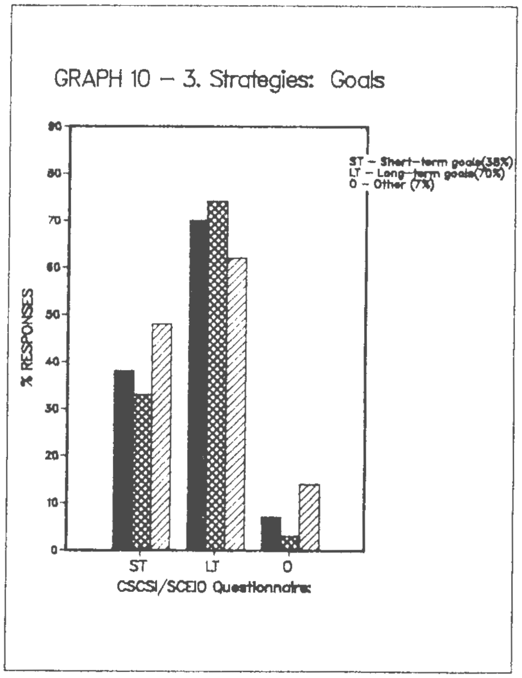




towards practical results was more important. As graph 8 shows, the strong feeling of academics and non-academics alike was that both directions should be pursued. Most seemed to feel that theory and practice are inextricably intertwined, with practical systems providing essential data for theoretical research, and theoretical results being extremely helpful to practical applications. As one respondent put it: "[you] can't separate them yet – [moreover, it is] undesirable to do so for political reasons"; and another noted that "practical applications offer a door opener and the possibility of support for more long term work". It was also suggested that many people still don't realise that AI is practical – "[we] need to get across to Canadian government/industry the idea that AI may actually be of use".

In the fourth question respondents were asked where they felt AI research and development could best be carried out (see graph 9). Substantial opinion (at 77 percent, the largest percentage in the questionnaire) held that universities were the best place (an opinion only slightly less strongly held among non-academics than among academics). The second most popular choice was industry. Many people opted for both universities and the private sector. One remark summed up this point of view accurately: "increased funding at all levels is crucial, building on strengths". Another remark indicates that industry should be more interested in AI research and development than it currently is: "[there is] a need to consider AI issues by industry and to remove the 'branch plant' R&D mentality". There was very little backing for research outside of academe and private industry, with government laboratories receiving only 13 percent support and such other suggestions as industrial conglomerates and private laboratories associated with universities receiving sporadic support. One person thought there was a need to avoid current organisational structures altogether, but made no alternative suggestions.

The second last question in this part asked about whether AI should set long term goals or short term goals (see graph 10). Academics (as might be expected) strongly preferred (74 percent to 33 percent) taking a long term perspective; but, somewhat surprisingly, non-academics also preferred long term goals to short term goals (62 percent to 48 percent). The argument for the long term view can be summed up in the following comments: "[we] shouldn't oversell the short term applicability – we don't have nearly enough knowledge to guarantee success in many, many areas"; and "[we] must recognize the extremely long lead time and considerable expense of doing AI research". However, it should always be realised that "AI is clearly a cutting-edge discipline with massive potential payoffs".



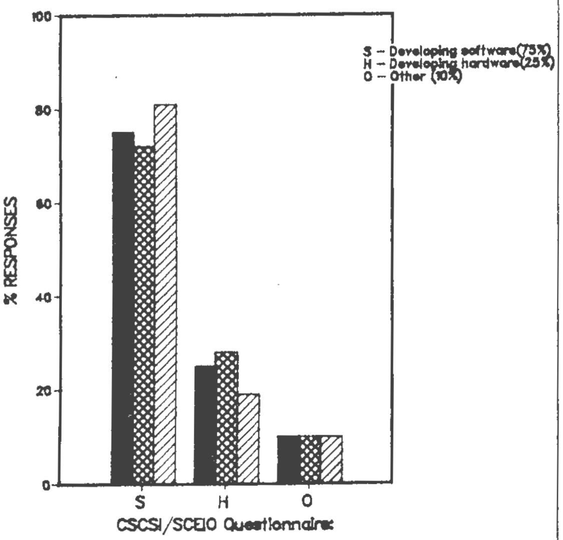
Finally, respondents were queried about whether AI should be oriented around software development or hardware development. Graph 11 shows a 3 to 1 plurality favouring a concentration on software, not surprisingly since only recently has hardware become a topic that has begun to take on any direct AI implications. The general feeling seemed to be that, in contrast to hardware, software development in Canada faces few competitive disadvantages ("[the] U.S. and Japan are awesome competitors in hardware - in software they have little or no advantage"); that "hardware is too expensive"; and that our competitors have too strong a lead in hardware ("[there is] no point in challenging the U.S. or Japan in [hardware] at this stage"). Backhanded support for hardware can be found in the following remark: "if [hardware] means LISP machines, no; if it means special purpose hardware for, say, vision, possibly". But, in general the most promising path seems to be software development (and as a few said "theory") rather than hardware.

Part 4 - The Role of the CSCSI/SCEIO

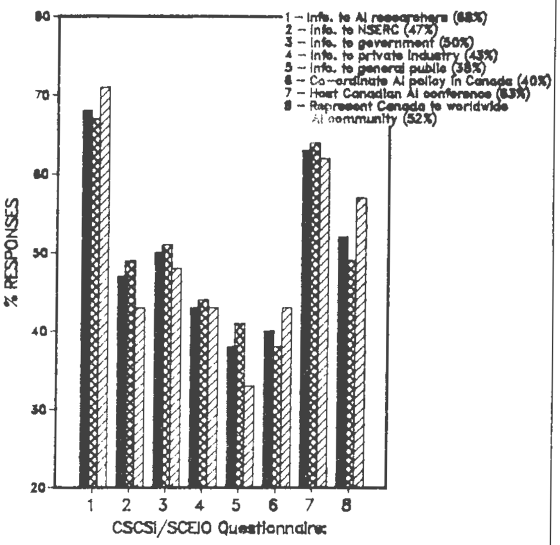
In part 4 of the questionnaire the respondents were asked to comment on the roles they saw as most important for the CSCSI/SCEIO, in order of priority. Some people did not indicate priorities; in such cases we decided to rank the choices in the order they appeared. This assumption does not affect graph 12, indicating the roles people mentioned anywhere in their choices, regardless of priority. Graph 13, indicating people's top three choices, may be a little less accurate due to this assumption, but since many people did not mark more than a few choices, this graph is not too far off.

According to graph 12, respondents support all 8 roles mentioned for CSCSI/SCEIO, with marginally more people in favour of its roles as "provider of information to the AI community" and "host of a Canadian AI conference" over its other roles. The other roles did not differ too much, with the least favourite roles, in order of disfavour, being "provider of information to the general public" (dead last), "co-ordinator of Canadian AI policy" (second last), and "provider of information to private industry" (third last). These preferences vary little between academics and non-academics, with slightly more non-academics being in favour of the CSCSI/SCEIO acting as the representative of Canada to the worldwide AI community and slightly more academics favouring its role as provider of information to NSERC. This is to be expected when it is considered that the non-academic category included all respondents living outside of Canada, and that NSERC is much more relevant to academics than to non-academics.

GRAPH 11 - 3. Strategies: Orientation



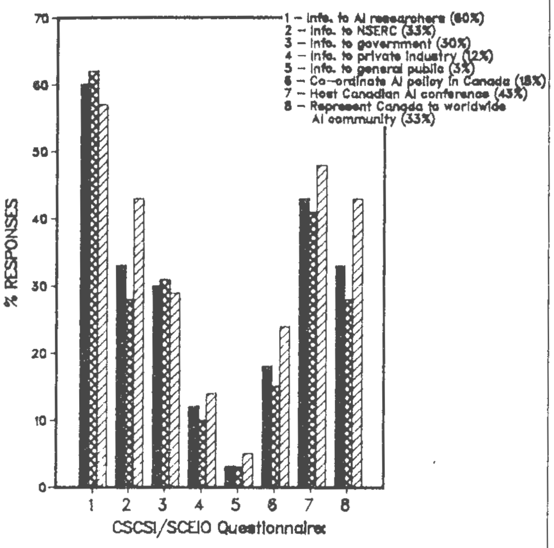
GRAPH 12 - 4. Role of the CSCSI/SCEIO (mentioned as important)



Graph 13 reflects the same trends as graph 12, but in a more pronounced manner. Due to our somewhat biased interpretation of the priorities, roles occurring later in the list should probably be boosted a bit and the earlier roles decreased marginally. Nevertheless, this graph is informative since it indicates strongly held opinions as to the role of the CSCSI/SCEIO. Unlike graph 12, this graph divides sharply into 3 equivalence classes of preferences. In the top class there is "provider of information to the AI research community", standing by itself (at 60 percent) as far and away the most preferred role. In the next equivalence class are 4 roles in the 30 to 45 percent range, in descending order "host of a Canadian AI conference", "representative of Canada to the worldwide AI community", "provider of information to NSERC", and "provider of information to other government agencies". In the bottom equivalence class are roles which range from 3 percent to 18 percent, in descending order "co-ordinator of Canadian AI policy", "provider of information to private industry", and "provider of information to the general public".

Respondents were asked to make suggestions about future roles for the CSCSI/SCEIO and many of them did. There were those quite satisfied with the CSCSI/SCEIO: "it is the only hope [for Canadian AI]"; and others were quite down on the organisation: "the CSCSI is 'mickey mouse', unprofessional ... I am not impressed with the performance of the CSCSI so far. AI in Canada is not synonymous with the CSCSI." But the general feeling was constructive: that we have done all right so far and can work together to effect the various changes

GRAPH 13 – 4. Role of the CSCSI/SCEIO (mentioned among top 3 priorities)



needed. The suggested areas for change were numerous. They can be categorised into 5 basic groupings: changes in the image of the CSCSI/SCEIO, changes in the publications, changes in the conference, suggestions for other roles, and changes in organisational structure. Each of these categories will be dealt with in turn, quoting liberally from the questionnaires.

The most numerous group of suggested changes pertained to improving the CSCSI/SCEIO's image in various ways. As one person suggested: "[we] must reach all of the AI community and win its confidence that CSCSI is a useful body". There were several ways of doing these put forward by the respondents:

(i) <u>improve public relations</u> –

> "[we need] more aggressive public relations work, i.e. when someone thinks 'AI' they should also think 'CSCSI'"; "generate lots of 'PR' to let the world know we exist"; "more press releases"; "[CSCSI/SCEIO] is not as widely subscribed to as it should be – an extensively advertised membership recruiting campaign is in order";

(ii) <u>influence government</u> –

> "[CSCSI/SCEIO should be] more active as spokesman to government" with the following ways having been mentioned: "try to get members into important positions in government and NSERC"; "[appoint within CSCSI/SCEIO] an official person or group to liase with and advise government"; "lobby in Ottawa";

(iii) <u>change the style of CSCSI/SCEIO</u> –

> "stop schoolboy responses – questionnaires"; "militancy"; "assertiveness"; "AISB [the British/European AI society] is successful – what has it got that CSCSI hasn't?"

(iv) <u>be more nationalistic</u> –

> "don't look so much to the U.S. AI community"; "decide whether [the] focus is primarily Canadian or international"; "should use Canadians more in the CSCSI";

(v) <u>foster better communication</u> –

> "[it] should be possible to get across to people on the fringe of AI what is actually going on"; "[CSCSI/SCEIO should] provide more information to AI people and related disciplines and find money/researchers in this area"; [CSCSI/SCEIO should] develop the 'administrative resources' to respond quickly to requests for information on AI in general and AI in Canada in particular"; as well as the comments noted earlier about the need for electronic communications links among the Canadian AI community.

The second group of responses generally relate to enhancements in the publications produced by the CSCSI/SCEIO:

(i) <u>improve the newsletter</u> –

> "[the] newsletter [should] be more frequent and regular"; "good explanations of research projects which don't read like grant applications or bits of Ph.D. theses might help outsiders get interested – it's that or Newsweek"; "[there should be] regular periodicals, reports, articles – aim at educated public and industry"; "[we need a] better newsletter – have a specific 'group' within CSCSI with more direct responsibility for producing the newsletter"; "raise the dues (if necessary) to support PR, a journal, and a <u>monthly</u> newsletter (a la SIGPLAN)";

(ii) <u>start a journal</u> –

> "[we] need an AI journal in Canada"; "[we need a] quick publication journal stressing tutorials and interdisciplinary research contributions serving to improve communications within the AI community, to involve scientists in other disciplines, and to establish visibility with the lay public";

(iii) <u>prepare a document on AI in Canada</u> –

> "[CSCSI/SCEIO should] put together a summary of AI (in a special CIPS mailing)"; "[CSCSI/SCEIO should] prepare a document based on members' opinions outlining the directions AI is taking and/or should take"; "some time consuming but useful activities might include our own view of where we see the field going ... [but] this is largely a matter of the time and energy people have to devote to the cause. I'd still rather see people do real scientific work."

(iv) <u>make use of electronic media</u> –

> several suggestions were made about having an electronic journal or electronic newsletter or electronic conference.

The third group of responses pertain to the CSCSI/SCEIO conference and improvements to it that would be possible: "a first rate conference seems a good way to guarantee credibility within the AI community (and I think this has been the case in the past)"; "[CSCSI/SCEIO should] make conferences higher profile"; "[CSCSI/SCEIO should] sponsor 'special topics' conferences (i.e. 'working' conferences) in off-years from CSCSI conferences, e.g. applications of AI to VLSI design, man-machine interfaces, etc." There was some concern that the CSCSI/SCEIO was being ignored in some AI conferences and workshops and should have a more prominent role in such affairs: "[we should] insist on [CSCSI/SCEIO] participation at all future conferences, workshops, etc., having to do with AI".

A fourth group of responses suggested other possible roles for the CSCSI/SCEIO. One of these was for it to act as a clearinghouse: "CSCSI should be a very competent, aggressive link between academia and industry, actively trying to set up pilot AI projects in industry or collaborative projects involving several AI centres"; "people in Toronto, Montreal, and Ottawa [could] promote CSCSI and act as contacts for anyone in their [research] area who needs AI information/referral/etc." There were many reactions to CSCSI/SCEIO acting as co-ordinator for Canadian AI, ranging from "no" to "that'll be the day". Another possible role for CSCSI/SCEIO would be to enhance its international outlook by representing Canadian AI abroad, perhaps, as one person suggested, by trying to gain "the right to send or elect a Canadian representative to the AAAI executive board", or conceivably by representing Canada on the IJCAI (International Joint Conference on AI) board if it re-organises along IFIPS lines. CSCSI/SCEIO was even mentioned as a possible source of funds to support AI research.

There was a fifth set of reactions that questioned CSCSI/SCEIO's current organisational structure in relation to achieving its goals: "[CSCSI/SCEIO] should review its relationship to CIPS"; "[we should] push for a true Canadian computer science organisation - I have nothing good to say about CIPS". Some people even questioned whether the CSCSI/SCEIΘ could (or even should) achieve its goals: "CSCSI hasn't enough manpower to fulfill all its goals"; "are there enough AI people to justify a separate Canadian organisation such as CSCSI?"; "the CSCSI is not perceived as a 'live' entity - apart from the conferences it doesn't 'do' anything - I'm not sure it really can. It's not anyone's fault." But, the majority seemed to be less pessimistic: "I think [the changes] have been happening - e.g. refereeing for CSCSI-84, generally better organization with continued effort and growth."

4. Some Important Questions for the CSCSI/SCEIO

The previous section outlined a number of possibilities for Canadian AI in general and the CSCSI/SCEIO in particular. In this section we discuss a number of open questions facing the CSCSI/SCEIO, questions which should be answered if the organisation is to find an appropriate niche in Canadian AI. Hopefully, the views of the membership as expressed in the survey and the feelings of the conference attendees will give guidance as to how these questions should be answered. Where we offer our own views, it is merely to give a basis for argument.

(1) The main roles currently played by the CSCSI/SCEIO are producer of a newsletter and host of a biennial conference. Can these roles be enhanced? In particular, should the newsletter come out more regularly? Should we sever the newsletter affiliation with CIPPRS and CMCCS/ACCHO? Should the conference be held annually? Should more money be risked on publicity in order to possibly gain increased attendance at the conference? Should "special topics" conferences be held? "Yes" answers to any of these questions will require more time and effort on the part of society members and probably more money from society coffers.

(2) Should the society consider taking on new roles? The current executive is trying to establish a new international AI journal under society sponsorship - will the increased profile this gives the society, and increased credibility, be worth the extra time, effort and "risk capital" expended? Are other roles possible? For example, could the CSCSI/SCEIO usefully serve as an information provider to government and industry, possibly by keeping the report "Directions for Canadian AI" up-to-date year after year? Should the society act as an accreditation agency for AI people? Again, any of these options would involve more work, and possibly imply an authoritarian role unsuitable to the organisation.

(3) Should the CSCSI/SCEIO change its orientation away from academic pursuits? It could host seminars on aspects of AI for government and industry. It could concoct policy documents on a wide range of AI issues for distribution to policy makers. It could more aggressively seek out funds, more aggressively advertise itself, and generally be more assertive in its demeanour. All of these options would change the fundamental character of the organisation, although many of them would enhance its normally meager coffers.

(4) Are structural changes necessary in the CSCSI/SCEIO? Specifically, should the society terminate affiliation with CIPS? If so, who would handle the many clerical functions provided by CIPS (handling membership lists, mailing things out, etc.)? Would independence free the society from any great restrictions imposed by CIPS? Other structural changes might be considered as well. For example, a publications officer with responsibilities for the newsletter and the journal (if it comes to fruition) might be a good idea. Should there be an executive member who is responsible for providing continuity in the conferences to ensure that conference sites are well chosen, ahead of time, to ensure that continuous publicity is provided, and to act as an information provider to each conference general chairman? More generally, should the entire executive serve longer terms in order to enhance continuity? Is the organisation too much of an oligarchy - should means be found to "open it up" to more participation? Our experiences on the executive convince us that it is very time consuming to maintain an awareness of all that is going on in AI in Canada, and that to do an effective job requires much thankless labour. We would be happy to have a veritable army of volunteers to help out, and any changes in the organisation which would help to spread the joy of working for the benefit of AI in Canada as widely as possible would be welcome.

(5) Finally, we must ask where resources to promote the CSCSI/SCEIO will come from? Although there are nearly 300 members, no more than 50 people are actively involved in the organisation, and many of them are ready to "retire" from active duty after service to the cause. Who is going to follow in their footsteps? The CSCSI/SCEIO is also ridiculously poor. The CIAR is able to find vast amounts of private sector money to fund its AI-oriented project; the Science Council has found money in the thousands of dollars for conferences and workshops; NSERC, the Science Council, MRC and the CIAR can invest enough money to help finance well over 100 people to travel to Ottawa. Why does the CSCSI/SCEIO have to struggle along with an annual budget of around 2000 dollars, with its executive members cleverly "finding" extra funds from their universities or their personal research grants in order to help finance CSCSI/SCEIO business, with its membership having to pay their own way to conferences and meetings, when there is obviously funding to support AI activities? Suggestions for how to increase funds flowing to the CSCSI/SCEIO are urgently needed.

There is currently an AI "policy vacuum" in this country. Government and industry are crying out for AI information and expertise. The CSCSI/SCEIO can act to fill this vacuum. It is the only Canadian organisation devoted to AI. Among its members are the most prominent AI people in the land. It has an illustrious history of service promoting AI in Canada. The CSCSI/SCEIO is well respected internationally. Through concerted action, the CSCSI/SCEIO can (and should) become a major influence in Canadian AI over the next decade, an indispensible asset in policy decisions affecting AI.

The society can, of course, choose not to become involved. Policy decisions will still be made. They will be made based on information provided by individuals who may or may not have the credentials to give the information. The decisions may not have the broad support of the Canadian AI community and they may not be communicated to the Canadian AI community. It is up to the membership of the CSCSI/SCEIO and its current executive to vigorously promote the organisation and its abilities in order that it cannot be ignored in the future. Never has the society been more relevant or needed.

Acknowledgements

We would like to acknowledge the other members of the CSCSI/SCEIO executive, Wayne Davis and John Tsotsos, for their support and encouragement (moral and financial) in producing the survey of AI in Canada. We would also like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) and Simon Fraser University through the Office of the Vice-President Academic and the Department of Computing Science for their financial support. Finally, we owe a debt of gratitude to the entire Canadian artificial intelligence community for providing their opinions on future directions for AI in this country.

References

1. G. McCalla and N. Cercone (producers), Directions for Canadian Artificial Intelligence, a report on the state of Canadian AI produced by the CSCSI/SCEIO, 1984.

The main body of the report gives specifics of AI research projects currently underway in Canada, details the concerns of AI people about AI policy and directions in the coming years, and discusses the history and current roles of the CSCSI/SCEIO. Its appendices include a directory of AI researchers in Canada, a list of expatriate Canadians involved in AI elsewhere than Canada, and a cross-reference list mapping areas of expertise to people interested in these areas. Copies of the report are available from Dr. Nick Cercone, Dept. of Computing Science, Simon Fraser University, Burnaby, British Columbia V5A 1S6 for $7 (Cdn) prepaid.

AUTHOR INDEX

Ali, Yawar.............133
Appel, Andrew..........129
Aref, Mostafa...........79
Atkinson, David.........73
Cercone, Nick..........167
Cohen, Robin.............6
Coles, David............99
Cox, Philip............138
Dean, Thomas............28
De Mori, Renato........103
Delgrande, James....87,162
Douglass, Robert.......159
Finin, Timothy..........25
Gershon, Ron...........133
Gilloux, Michel........103
Glicksman, Jay..........44
Hamey, Leonard.........129
Hegner, Stephen........159
Hirschberg, Julia.......11
Hummel, Robert..........58
Howe, Adele.............25
Jacobson, Guy..........129
Jenkin, Michael....106,133
Kant, Kamal.............62
Kay, Martin..............1
Kramer, Bryan..........153
Langley, Pat............95
Leclerc, Yvon...........50
Letovsky, Stan..........70
Levesque, Hector.......141
Mackworth, Alan........114
Mauldin, Michael.......129
Mercer, Robert..........16
McCalla, Gordon.....79,167
Mokhtarian, Farzin.....114
Pietrzykowski, Tomasz..119
Pollack, Martha.........20
Poole, David...........124
Rendell, Larry..........99
Rosenberg, Richard......16
Sage, Stephanie.........95
Salzberg, Steven........73
Shafer, Steven..........33
van Emden, Martin......126
Zucker, Stephen...50,58,62